MASTER IN COMPUTER SCIENCE

Software issues for software products

Pouplard, Quentin

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique.
Année académique 2003-2004

Software issues for software products

Quentin Pouplard

Mémoire présenté en vue de l'obtention du grade de Maître et/ou Licencié en Informatique.

Abstract

Security seems to be an important concern for a growing number of applications. This document intends to clarify this idea by analysing the market situation regarding security. This analysis is done from the buyer's (and user's) point of view, but also from the seller's (and developer's) point of view. Two aspects are analysed: the economical aspect and the technological aspect.

The second part focus on the exploration of a development methodology for securing application (the ISO norm known as "Common Criteria") allowing developers to formalize the security needs and buyers (users) to get a better idea of how secure a product can be. The main purpose is to get more secure applications and more structured informations regarding an application security. An evaluation of this methodology is made to discover the advantages and potential drawbacks bound with its application.

La sécurité semble poser un problème important pour de plus en plus d'applications. Ce document tente de lever cette incertitude en analysant la situation du marché par rapport à la sécurité, aussi bien du point de vue de l'acheteur (et des utilisateurs) que des vendeurs (des développeurs) et ce d'un point de vue économique et technologique.

Puisque que la sécurité est nécessaire, la seconde partie se consacre à l'analyse d'une méthodologie de développement (La norme ISO connue sous le nom « Common Criteria ») permettant de formaliser les risques et les besoins en sécurité d'une application. Ceci afin d'obtenir des applications plus sûres et de permettre au client d'obtenir plus d'informations sur la sécurité de son application. Enfin une évaluation de cette méthodologie (sur base d'un exemple) est faite afin de découvrir ces avantages ainsi que les problèmes potentiels liés à son application.

Acknowledgments

To Professor Jean Ramaekers for the supervision of this work.

To Patrick Jacobs, one of the director of Océ Software Laboratories, for his help integrating the company.

To Frédéric Rouyr, Jacques Flamand, Etienne Goublomme and the others employees of Océ Software Laboratories for their help, advice and kindness.

To my family and friends for their support, help, spell-checking during the writing of this document and for helping me going trough those five years of studies.

To all the professor and assistant from the informatics institute of the university of Namur for everything I've learned during my studies.

Glossary

Assurance — Grounds for confidence that an entity meets its security objectives of one or more assurance component(s) from Part 3 to an EAL or assurance package.

Class — A grouping of families that share a common focus. In this document, it refers to functional requirement families.

Component — The smallest selectable set of elements that may be included in a PP, an ST, or a package. It can be an objective, a threat, a policy, and assumption or a functional requirement.

Evaluation Assurance Level (EAL) — A package consisting of assurance components from Part 3 that represents a point on the CC predefined assurance scale.

Family — A grouping of components that share security objectives but may differ in emphasis or rigor.

Protection Profile (PP) — An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.

Role — A predefined set of rules establishing the allowed interactions between a user and the application.

Security attribute — Information associated with subjects, users and/or objects that is used for the enforcement of the application security rules.

Security Function (SF) — A part of the application responsible for a specific security related functionality.

Security objective — A statement of intent to counter identified threats and/or satisfy identified organization security policies and assumptions.

Security Target (ST) — A set of security requirements and specifications to be used as the basis for evaluation of an identified target of evalution.

Target of Evaluation (TOE) — The application that is being taken in consideration. The most general definition include any kind of system, but this paper limit this definition to software application.

TOE Security Functions (TSF) — The set of Security Function of a given TOE.

TOE Security Policy (TSP) — A set of rules that regulate how assets are managed, protected and distributed within a TOE.

TSF data — Data created by and for the TOE, that might affect the operation of the TOE.

XML — eXtensible Markup Language, a standardized open markup language designed to help in the exchange of information.

XHTML — and "XMLized" version of HTML. It looks like HTML but has a more strict syntax. The main interest of XHTML is that it is a specialization of XML therefore it can be processed by XSLT processor.

XSL(T) — A programming language using the XML format designed to take an XML file as an input and produce some kind of output (an XHTML file most of the time). This language is widely used to separate the logic and visual part of an application.

Table of contents

1 Introduction

Security is a very important problem in today's applications. Because the subject is rather large and cover lot of aspects (management, development, marketing, cryptography, etc.) this document will try to approach cover two aspects of security in software application from two points of view.

The first question when approaching security is "Is security actually needed?" To answer this question we need to focus on the two sides of software life: the creator's side and the user's side. The creator of the software is responsible for mainly two things: developing the software (including the maintenance) and selling the software. The user has to buy the software first and use it. The first section will try to determine if security is needed for the two parts of each side: users, consumers, developer and software's seller. The conclusion of this part is that security is relevant for every intervenient at a different level: security is an implicit feature for the users. The buyers want a good product so the need of the users must be fulfilled by the software company. At the end of this chain, developers need to implement software that is sellable. Apart from this economic point of view, this section also develops the fact that secure software is bound with the notion of "correct" software from a technical point of view. Because the answer to the question is a big "yes", we need to perform two more steps: explore the actual problem in order to be able to express it in a more formal way (see 2.3 Coverage) and formalize a way to make more secure software. This last step will be exposed in the second section.

The second part of the document will try to formalize security in software development, starting from the threats partially exposed in the first part; we'll try to expose some way to respond to those threats. This section is based on the existing criteria on security: Common Criteria. Common Criteria defines two types of documents Protection Profile and Security Target. After a brief history of Common Criteria, we will develop the way to build those two documents.

The third part is a presentation of a sample protection profile built for Océ market: "outputware". After a description of "outputware" a sample Protection Profile is exposed and some evaluation is done on the protection profile based on the sample PP writing experiences.

Because security is a very large domain, the conclusion will be in two parts: conclusion of this document and information about some possible future works on security to continue the ideas behind this text.

2 Context

This section aims to limit the scope of this document. First of all, the document focuses on the definition of a "system", and tries to determine how this system is relevant for developers and consumers. Then we will point out the origin of security: threats. To end this section, we derive some objectives and security levels from the possible threat.

2.1 Overview

Security does apply to a large domain, from the smallest embedded system to the biggest mainframe. In this point we are going to expose which kind of system we will focus.

In some words: "system" in this document is related to "a desktop or server computer running software's and possibly exchanging information on a network". We don't consider embedded system (some of them could be seen as a subset of the above definition) or system having direct mechanical interactions with their environment. This document is about software, not hardware or human related security issues.

By limiting the scope of this document, we don't mean human related security issues are not important. A common principle in computer science also applies for security: "the weakest link". Your system can be totally secure, the problem remains: you can ask 20 passwords before reaching a document, once it's printed, nothing but you can avoid it to fall in the wrong hands. And that's also why security is so hard to achieve, everything can be perfect except for one single line of code, and your system won't be 100% secure. Moreover if this single line is abused, you may encounter the same loss of benefits as if the whole software was flawed. Therefore the objective is to achieve correct security, trying to reach perfection and knowing it's impossible.

2.2 Motivation

Now that we know what kind of system we are talking about, we can begin with the unavoidable question: "why security?" First we try to see if security makes sense from a technical point of view then we focus on economic reality in order to get some realistic idea of the "interest" opportunities of being secure.

2.2.1 Technical

From a technical point of view, security can be seen as "a way to make a system do what it's intended to do ... *and not more*". Making system "do what it's intended to do" is simply the respect of the various specifications, but it doesn't guarantee that we won't be able to do more. The point is that this "*more*" can be things like injecting and executing code into one application. This is of course pure theory, but the point is there: making secure software offers some guarantees that specification will be respected even when the system will be misused or stressed.

In fact, making secure software is some kind of defensive programming: we don't count on things to work well; we just expect anything to happen. So we will design our software to handle any kind of situation in a correct way. It's not a full pessimistic approach however. Due to reality constraints, software is not able to assume all the possible cases. However, they have to fallback correctly in case of a problem or misuse, and they have to do it without any corruption. For instance, the implementation of rollback procedures allows us to implement "correct" behaviour. If something goes wrong, returning to a "clean" state, we do not actually handle the specific error or attack. Nevertheless the result is that our software is still in accordance with an acceptable way of working.

Another advantage of secure software is that it forces some restriction on development: any line of code might be a security hole, so developers are obligated to take care of what they do. It also constrains a developer to think about the impact of implementing this "new great killer feature", the impact on his code, on other people code, and finally on the whole system. This consideration can be extended to other aspects: if you care about memory management, you're forced to pay attention to memory allocation; if you care about speed, you're forced to pay attention to anything that can be CPU bound. For security, you'll just need to pay attention to ... almost anything. In fact, it even has an impact on memory management (buffer overrun/overflow) and speed optimization (removing test to gain some CPU cycles).

Of course, this is theory, no economic reality in there. The truth is that security (like any other development activity) has a cost. As exposed in previous paragraph, perfect security isn't likely to be achieved, so it could become an endless financial hole for a company. The next paragraph will try to put some reality in there: from a technical point of view, security is "good" for your application, what's really good for your clients?

2.2.2 Market

This section will try to answer to the following question: "Is security relevant for consumers?". In other words, do software buyers want security, and most of all: will they pay for it?

Before investigating this question, another point may occur: "is security relevant for your company?" The fast, rough answer is no! Security has a cost, it takes time and it can even influence the project development's method (see 3.7.2 Evaluation Assurance Level (EAL) and other assurance related classes). But these facts can't be satisfying. First of all, the technical point of view gives us some idea that developing secured software can lead to better software (and possibly less support cost, integration cost …) and the second point is obvious: if security is a feature required by your client, then security is relevant for your company. That's why it's fair to take the client as the basis of our investigation: the needs of the client are (or should be at least) the needs of an enterprise, as long as this enterprise needs to survive and make profits.

a. First sight

To determine how security is important to clients we can watch some advertising, press articles, news, etc. After a first sight, one thing is sure: people talk about security, but the meanings of information they get/publish depend of the kind of users, we can separate them in 3 classes described in the following table.

Home end-users	This category represents anyone using computer outside his company. You get information about security everyday: *"It's 11:54am and I've about 10 Spams about security related issue in my mailbox everyday. Today, the Microsoft website headlines 3 step to protect your computer. As everyday,*

	securityfocus.com is active (7 vulnerabilities in various softwares today). Last week, 3 security related patches were published by Microsoft, and at least one critical for Linux/Unix software." Therefore, you can come and get information, or the information comes to you in form of Spam or automatic update. Selling security is easy for at least one reason: people's fear! It is easier to sell something when you can make people believe that their computer will stop working or that other people can steal them. It does not mean that security is not useful, but it means that you do not need to run and pay for everything to be correctly safe. The same kind of things happens with many situations of the everyday life: a good lock is great for your main door, but you won't always buy the most expensive one, and adding two other locks won't be useful if the first is efficient, it will just be a bit more boring to close/open your door. Those users are interesting, and represent an important part of the PC users, but we will not focus on them. It could be the topic of other documents like "how to make family pc more secure?" and seems to be at the core of some marketing operation from major software vendors.
"Geeks"[1]	They can be seen as a subclass of end-users. They are most experimented and interest in computer. (They are talking about how "xyz" 1.02 is worst than 1.03.) They are often able to accurately identify differences between two releases of software. Tracking the last security flaw of products, some of them are even the pirates. This category includes the most stupid "script kiddies" up to the best hackers. The main point is that they have some knowledge of

[1] You can find many definitions of « geeks », basically it's someone who spend time on his computer for « fun », you can get a more complete definition on http://www.circus.com/~omni/geek.html for instance.

	their system, and have some possibilities to abuse other's systems.
Company/corporate users.	That's the most interesting one for us. First of all, company employees and end-users use the same kind of hardware/software; .Subsequently they are concerned by identical kind of threats. . However, it must be thought differently: an antivirus software for the whole company and your personal home computer one' are rather different even if they use the same technologies. Loosing a personal document with your pet's name or losing a document targeted as internal use in your company isn't really the same, the effects are rather different (well, except if your company sells pets…). For those reasons, one can understand that advertisement for security targeted to company is distinctive. You can find some on the web, in dedicated newspapers. For now most of the advertising is third party product that try to add more security to a current product (adding firewall to your OS, antivirus, safe backup, etc…). As a side note: One of the most widespread security related software seems to be secure payment framework. That sounds logic, money is money! People realize that something is critical if money is involved, payment has always been a problem during transaction, even before computer and the internet.

From those 3 categories, we will only consider one: corporate users.

b. "Outputware"

As we will focus some part on this document on "document management" software[2], we will have a look at security's advertisements in output management products. Many companies are working in this domain of application; the following will be based on seven of them.

The first step of the analysis was to search the product's information, to discover how they advertise "at first sight" the security of their products. The next step was a browse among(Google search on) their websites to get how many pages contain the word "security" This count down aims to give an idea of the amount of information about security that you can find. The last step was a global Google search with their product names and the word "security", in order to get external information. The results were rather poor so additional searches have been made using "common" security related words, in order to find more information. It wasn't really a success neither, and didn't provide more information. Remember that the purpose was to see how the security is advertised, and was not about finding information on security on the whole Internet.

Some researches have also been done from the developer point of view by reading some schedules. Nothing relevant has been found, unless the product was dedicated to security none of those document focused on security. It was just implicitly secure.

Name	Website	#result[3]	More information
SEAL systems	http://www.sealsystems.de	13/7	No real security concerns, their products are just "reliable" and Google gives 13 results with security on the main website. (with lot of off-topic entries).

[2] Basically, document management software are software that take many kind of input and route them to many kind of output. Generally, document are the input (windows printing services, SAP, ...) and printers are the output. The role of this software's is to add transformation between and interpretation (accounting, reformatting, enhancement) in an efficient way (using load balancing, advanced queuing, etc...). See 4.1 Usage model for a detailed explaination.

[3] Results are the number of result found by the Google search engine. The first number is the number of result exposed by Google, the second is the number of relevant result (sometimes the word security appears in a webpage for a different reason than product security, for instance web browser security settings needed to view the website).

Dazel (HP)	http://www.dazel.com/	3/0	No real security concerns, it just "ensures security". No off-topic documents, but no page are about security, it's always in context of kind "necessary security", "ensure security".
Agfa delano	http://graphics.agfa.com/product/CatProd_DisplayPublic.html?id=7539	1/1	No security concern, their software is just reliable. Google gives one off-topic result. External search gives 2 off-topic results. Security really doesn't seem to be an issue for this solution.
IBM	http://www.printers.ibm.com/internet/wwsites.nsf/vwWebPublished/ipmgraixhome_eu	32/2	No real security concerns, again, their products are reliable.
Kofax	http://www.kofax.com	~89	They have something about security, 89 results in Google, a question in the FAQ. But they rely on third party for most of their security. "The software is secure because the environment is secure"
Océ	http://www.oce.com	95	More information about security (security mixed with confidentiality becomes a feature), some misleading between "authentification", "right management" and "security".
Sharp	…	148	The website seems to concern only software related to hardware product (drivers, bios, …) The results are for all the Sharp products.
In order to make some comparison, here are some other results:			
Microsoft	www.microsoft.com	~139.000	
Microsoft Office	http://office.microsoft.com	~609	This result is higher than what we've got for "outputware", and it doesn't include developer related information (http://msdn.microsoft.com), supports (http://support.microsoft.com) or beta products (http://beta.microsoft.com).

From this investigation, we can extract the following information:

- Security doesn't seem to be one of the primary objectives, like if it was "secure by default", or "implicitly secure". Why that? If security doesn't seem to be advertised as an important feature, does it mean that it's not a decisive feature for the client?

- When the company advertises security they often rely on the third party software they're using for security concern ("use internal IIS security", "rely on Windows 2000 user management", "based on the proven Unix technology"). It can be compared to table in Annex A. showing that no system (OS) is completely safe. A basic rely on third party application for security doesn't guarantee anything.

- That seems logic that Microsoft does advertise more about security in his product: it's a bigger company, much more of software's (but when we consider only Microsoft Office it still gives more results that all the "outputwares" combined), and other company relies on Microsoft for security. Another fact is that Microsoft isn't very famous for his security, that shows us that advertising security and being secure isn't the same.

c. Some facts

To think about this problem, we have to take the specific aspects of security in account: security isn't really a feature, it doesn't add anything to the application (it can impact some of the feature, like the need to enter user and password to use the software, but by itself, it's not a new functionality), it just ensures that the application runs as expected, even when someone try to abuse the system. It could be seen as "implicit": when you buy a car, you're not looking at the feature list to check if it's able to run; it's a car, without that it can't be called a car. It seems that it's the same for software: when you buy software, it should work, if not, it can't be called software. Claiming that users want their application to work seems obvious, and it is! From this point we can try to determine if security is really an issue in our current world: how fast will an application be broken, if it's not secure? There is no answer to this question, it depends of lot of factors, but it can be interesting to look at some facts[4].

From ZDNet, 29 January 2003, http://zdnet.com.com/2100-1105-982554.html

PSINet Europe purposely built an unprotected server and connected it to the Internet to determine how quickly it would be compromised. Their findings were astonishing:

- The server was maliciously attacked 467 times in the first 24 hours

[4] We must take those statistics with care: most company doesn't want people to know they were successfully attacked. It's not really a problem here; we can take that information as a minimum. Because of the large number of result of some query, off-topic search wasn't done for every search.

- Most of the attacks originated in the US or Western Europe
- After 3 weeks, a total of 626 attacks were detected against the server

This experimentation gives us important information: the threat exists, your system can be the target of pirates, they exist, they're there, and they may try to abuse your system. It doesn't mean that your system will suffer 626 attacks in 3 weeks, but the weakest link is still true here: one successful attack can do the same damage as 626 attacks. It's not something that happens only in movies. (In September 2003 the source code of a future game from Valve software was stolen and made available on various IRC and P2P network.)

From CERT/CC, 17 August 2000, http://www.cert.org/present/cert-overview-trends/index.htm

Carnegie Mellon University estimates that 99% of all reported intrusions "result through exploitation of known vulnerabilities or configuration errors, [for which] countermeasures were available." This directly shows how truly important it is to regularly patch systems, as well as keep current with network and system countermeasures.

In a test to see how fast a non-published, unpatched, system would be discovered, the San Diego Supercomputer Center placed a default installation, Red Hat Linux 5.2 machine on the Internet.

- 8 hours after installation, the system was probed for RPC vulnerabilities.
- 21 days after installation, there had been 20 targeted, unsuccessful, exploits attempted.

Approximately 40 days after installation, a vulnerable POP service was compromised, and the intruder installed a Sniffer, several backdoors, and wiped out the system logs.

This one add one important information: 99% percent of the attacks were "avoidable attacks", and that's a very important point, still the weakest link: the latest version of your software can be free of (known) security hole, if your users are still using an old version, their systems aren't secure, if they misuses your software, misconfigure something, doesn't protect access to critical data, their systems won't be secure. It can lead to two partial conclusions:

- Software update is important, if developers could make update easier, they would make their software more secure in the real world, so software update is a way to get more security, making your software easily updateable is important for your users. Another important aspect is ease of configuration of security related features. This aspect seems to be the target of some software company; it's a big improvement for security.

- Users and administrators need to be trained! Software applications are like lot of other goods: they won't run for year without update. It seems obvious... but it looks like it's not really applied. Lot of users intentionally turn off all the "automatic update", "virus protection", etc...

From Computer Economics, 2 January 2002, http://www.computereconomics.com/cei/press/pr92101.htm

It is estimated that the worldwide impact of malicious code was 13.2 Billion Dollars in the year 2001 alone, with the largest contributors being SirCam at $1.15 Billion, Code Red (all variants) at $2.62 Billion, and NIMDA at $635 Million.

This one gives us an estimation of the cost impact of malicious code (virus), of course it's "estimated" and this kind of information is often to take with care. But even if the result is biased, it's about billion dollars, not some thousand dollars; it can give an idea of how far an insecure system can generate costs.

From Joint CAIDA, ICSI, Silicon Defense, UC Berkeley, and UC San Diego, 01 February 2003, http://www.caida.org/analysis/security/sapphire/

An analysis of the Sapphire/Slammer SQL worm shows:

- "This worm required roughly 10 minutes to spread worldwide making it **by far the fastest worm to date.**"
- "In the early stages [the number of compromised hosts] was doubling in size every 8.5 seconds."
- "At its peak, achieved approximately 3 minutes after it was released, Sapphire scanned the net at over 55 million IP addresses per second."
- "It infected at least 75,000 victims and probably considerably more."

From
http://www.deloitte.com/dtt/cda/doc/content/Global%20Security%20Survey%202003.pdf

From this study, we can get some pieces of information:

- 63% of the enterprises see security spending as a "necessary cost of doing business".
- About 39% of the respondents acknowledged that their systems had been compromised in some ways during the last year.

This study can help us understand how companies see security, for most of them it's "necessary cost of doing business". It enforces our idea of security as being implicitly needed: it's something needed to run, not a feature that you'll prefer over one other when you're looking for a solution.

We could also wonder if those kinds of attacks are new, if security problem are new, media often present security threats as "new", "caused by the internet". It seems logic to think that more and more connected architecture can lead to more attack, and it must be partially true. On the other side, we can find information about number of "computer related crime", for instance in 1976; one source identified 339 cases with an average lost of $544.000[5]. Most of them were simply employees modifying their financial records.

2.2.3 Conclusion

From all those information, we can extract one fact: security is a concern for the client, not a direct feature request, but something implicitly bound with their idea of software. They want software that runs, and if the software contains holes, there is a potential risk of failure. It's not new and probably won't stop during the next years.

2.3 Coverage

What does security implies? What kinds of things are important for a secure system? We can extract some relevant point from the above information:

- A large part of "attack" comes from the inside.
- 99% of the attacks are "known vulnerabilities".
- It can go very fast and be very expensive.

[5] It's also important to note that the median loss is very close to this, it's not biased by one or two important case around hundreds of small one.

The fact that "attacks come from the inside" lead us to think that people play a very important role in security: always the weakest link, your printer can be as secure as possible, if someone steal the printed document, security isn't achieved. The same kind of things applies for all security related domain (your password can be as complex as needed, if there's a post-it at the top of the screen, well it's "just" a reminder...). Peoples also play an important role in maintaining their software: if they don't apply patch from software vendor, their systems can't be secure, even worst: known security issue are easy to find on the internet, hacking a system becomes as easy as "googling" for the last news about your favourite artist[6].

Covering those kind of domains can be very interesting, but is more related to internal organization, ease of use and isn't directly related to computer science (some of the rules that can be applied could also apply in an environment without computer, things like maintaining a list of people allowed to enter a building, to access a resource, etc...).

There's also lot of other aspects that could be interesting to cover, like legal issues: how far is it allowed protecting their data, can we make them unreadable by legal authority? Can our boss read our e-mail? Or is it legitimate to protect them? What kind of cryptography system could be used? Can audit trace be used as evidence in a court? ... Again all those question can be very interesting, but it's more law than computer sciences.

We need to focus on computer sciences, and even more than that on software. When talking about security, most sources isolate 5 main security features (developed in the next sections); that doesn't mean that the other doesn't have anything to do with security, but it's a good start.

[6] A big distinction must be made between cracking the system (for instance: get more right, read a confidential document...) and no being identified. The first part is easy, the second part becomes more and more difficult. Being anonymous on internet is one of the real issue about cracking a system.

2.3.1 User authentication and authorization

User identification is the basis of security, to protect something against someone; we need to know this someone (authentication). Applying a "no access at all to anybody" isn't acceptable, it's rather obvious: if we need to protect something, it's because it has some value to someone, and this someone will of course need to get this information in an understandable way. Even with simple systems with only one possible user, this user, even if unique, is someone, can and can't do specific actions on the system (authorization).

2.3.2 Data protection

Data protection covers all the way software can use protect their data's, this include of course data encryption. It's heavily related with user identification and encryption key management. The purpose is of course to disallow unauthorized people to retrieve information from the protected data.

2.3.3 Communication

Communication covers the exchange of information between only subsystems of our system. Secure communication is of course need for data protection, and proper user authentication. This topic will focus on how to ensure "safe communication". This rather wide topic is covered by some "common principles", like the ACID[7] nature of transaction.

Communication is separated from the rest because it has some specific issue, for instance a secure, encrypted system with safe user identification could be abused with things like replays[8], repudiation, etc.

---

[7] ACID: Atomicity, Consistency, Isolation and Durability. Four characteristics needed to provide reliable transaction in a distributed environment.

[8] Replays basically consist of repeating the same date twice on a connection, with or without small modification to get the expected result. For instance repeating two times the same bank transaction can make the pirate twice as rich as expected.

2.3.4 Audit

Auditing is important and very important when security isn't efficient! That may look a bit strange, but starting with the fact that no system can be 100% secure, having good logging of what happens to the system can provide two things:

- Being able to recover faster from an attack (the pirates touched this document and this one only, this kind of things).
- Being able to get more information about the attacked system (which IP, when, how many times, ...). Most of the times, this kind of information that makes authority catch the cracker.

That's again one of the important facts of security: nothing is 100% secure, everything can be abused, making software secure sometimes implies taking care of abusing, making things perform better if it goes bad. A correct auditing of a system could of course help prevent attack ("someone is trying to brute force my password, I'll stop him") but also help after the attack!

This domain also has some legal issues, for instance what's the value of this kind of information in a tribunal, and can a trace be used to make someone guilty or innocent?

2.3.5 Key management (cryptography)

Cryptography is the key to protect data; if existed before computer existed[9]. It's a huge domain covered by thousand of books, articles and it could lead to thousand works like this one. That's why we will focus on a very small subset of cryptography: key management, and even smaller: not how to store the key safely, but only the management of the key as critical resources. That can be seen as data protection but not related to the application "useful" data, but to the "security related" data.

[9] One of the first known forms of cryptography is the Caesar Cypher which is said to have been used by Julius Caesar to communicate with his army. It's a simple substitution cipher consisting of shifting letter in a message.

It can be useful to note that one of the domains that have the most legal issue is cryptography: most of the algorithm used comes from military sources; some country doesn't allow the use of cryptography (or the use of cryptography stronger than a given level).

2.4 Threats

A system is not secure by default, specific action need to be taken in order to improve the security of a system. Even worst: perfect security can't be reached. We will try to determine the kind of threats that may affect a system.

Focusing on threats can be very long and difficult, that's why we will only focus on threats related to software, not their meanings. For instance data destruction can threaten lot of part of a company, it can lead to loss in various domain for the company itself and for their clients, but no matter the real threat it represent, for the software part of the system, it's still data loss.

2.4.1 Data related threats

Data related threat are too wide, because computer science is often described as "information management", and because information is represented in some kind of binary format often called data, anything that can affect a system is "data related". This section won't focus on that, but only when pirates directly affect the data: destroy data, modify data or steal data.

a. Data loss

Data loss is our first threat, the most well known kind of data loss is web defacing, every day, some website are defaced by hackers for various reasons, the basic idea is to destroy the current website content and sometimes to replace it with another. Of course, it's not limited to this kind of loss, it can go a lot farther and sensitive (non public) information can be destroyed too. The idea beyond is always to destroy something, not to use them or take any profit of the information (but one can get profit from the loss of another…).

b. Data corruption

The second threat is data corruption, pirate modifying sensitive data to make them have another meaning. This can be achieved directly (by editing files for instance) or indirectly (by modifying the way the system works, for instance redirecting any mail to a given mailbox ...). It's important to make the distinction between data loss and data corruption for the following reason:

- Data loss is visible: the data isn't there anymore. Data corruption can take long time to be discovered. For instance an attempt was made to corrupt the Linux kernel source tree. Fortunately the error was discovered by watching CVS[10] logs.
- Data loss is "easy", hackers doesn't need to understand what the data are.

c. Data stealing

The third data related threat is steal of information, hacker doesn't modify or delete anything, they just the information they need, and it can be any kind of information, from the source code of the next FPS[11] of Valve studio[12] to a series of credit card number stolen from an e-commerce website[13] or other things like a internal Microsoft memo about future marketing policy against Linux[14]. It can of course also be any kind of industrial or intellectual property.

---

[10] CVS: Concurrent Versionning System: Software that help teams of developer manage a single source repository. A CVS server audit all the operation made on the repository. This example is also a good illustration of why auditing is an important security feature.

[11] FPS : First person shooter, videogame where the view of the game try to look like what a human see, the player is « inside » the body of the people he is controlling. The first FPS was wolfenstein in 1990. This kind of game is also called « Doom like ».

[12] See http://www.gamespot.com/pc/action/halflife2/news_6076314.html for more information.

[13] See http://money.cnn.com/2003/02/18/technology/creditcards/ for more information.

[14] See http://www.opensource.org/halloween/ for more information.

It's important to note that, if the first threat can be easily detected (after the attack), the two last can only be efficiently detected if a good logging is performed on the system.

2.4.2 Availability

The idea behind this is to make a service unavailable in one way or another, it can be seen as a kind of data destruction, but generally data are just made "unavailable", and not erased.

There are two widespread way to make a service unavailable:

1. "Crashing" the system.
2. "Flooding" the system.

The first one generally implies to use the system in a way it was not designed to. For instance sending him email addresses of 500 characters where the system only indented to get a maximum of 256 characters. If the system isn't build against this kind of attack, it may crash, and thus it won't be able to provide his services anymore.

The second one is more complex to put in place and more complex to prevent (almost impossible). The concept is very simple: send million request for a service until the system doesn't have the time to answer. The big problem with this kind of attack is that it's simply using the system in a "normal" way and during this kind of attack it's very difficult to make the distinction between "good" connection (from a real users that really need to service) and "bad corrections" (intended to flood the system). The distributed nature of those attacks makes them hard to detect: lot of computers are attacking a single target, blocking a single IP can't do the job. Lot of recent virus are doing this kind of DDOS (Distributed Deny Of Services) attack, the most known was blaster, a virus designed to flood one of the Microsoft Windows Update website (http://www.windowsupdate.com). More recently (may 2004) a new virus was created from a flow in another part of the Windows operating system, the virus is called "Sasser".

2.5 Objectives and security level

From the threat of the section D, we can extract the basic objective of a secure system:

- Enforce reliability and system availability.

- Ensure data protection
- Enforce confidentiality.

To reach those objectives there is a set of tools and concepts (encryption, transaction, role based user management, journalized file system…) and that need to be put in place by developers to be efficient. Next section will try to dig a big deeper in this topic, mostly how they can be expressed in a clean normalized way.

On the other side and because a 100% secure system can't be reached, there is some way to make claims about a system. "Security levels" are important in order to get more information about the security that is really implemented. Security level goes from the lowest to the most proven. There are different scale that exist (EAL used in this document but also respective scale for each country-specific criteria, see 3.7.2 Evaluation Assurance Level (EAL) and other assurance related classes) but they can be summarized as: At the bottom, the less secure, kind of "our product is secure because we've tested it" (functionally tested), and the top, a system that is "proven to be secure, tested by a third party with a given method" (formally verified design and tested).

3 Model

Security is a never reached objective: achieving security has an impact on all the development of an application. For all the task related to development, some model, some methodology exist to help us, that's the same for security: there are some step that can be achieved to give some structure to our security needs, to be able to present an application as secure (or not) more easily. We will focus on that for the next part of the document, we will present a model that try to help us in our task.

We will use a model called "Common Criteria", to be more precise the "Evaluation Criteria for Information Technology Security". We will try to explain why this model was chosen. After, we will try to discover for whom this model can be useful. From there we will be able to dig into the model.

But first of all, because common criteria, like lot of model, imply the use of dedicated vocabulary and concept, we will expose a fast presentation of common criteria, in orders to "get into the concept behind" common criteria.

3.1 Fast presentation

We know, we need to formalize security requirements, to achieve this objective, Common Criteria use a threat/objective approach: attack against a product represent a *threat* (see 3.5.2c.) for a company, therefore for the software they use. Companies also often have some "rules" to follow, external to the application, for instance, if a company want ISO certification, some rules apply (called *policies* see 3.5.2b.). Also, we live in the real world, so we can't expect too much from security: it must be realistic, realizable; therefore we also need some *assumption* (see 3.5.2a.) about our environment, to filter what's realizable in our context and what's not (for instance, any password can be abused, it may takes time but it's still possible, so a "unbreakable password" isn't realizable).

Our security needs come from these 3 things: threats, "company rules" called policies and real world constraint called assumptions. Those 3 things allow us to derive some objective to fulfil threat, policies and assumptions.

And finally from those objectives we will be able to create the requirement list (see 3.5.4 Functional requirement), which will be the basis of our system's specifications.

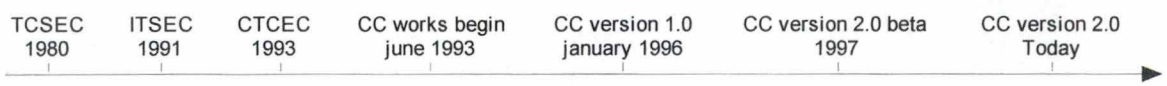
Those simple concepts allow us to understand the interest of common criteria: structured information. With such a structure, we are able to define some rules to apply in order to test if our criteria are coherent, correct and in some way complete. That's the second part of common criteria, assurance evaluation (see 3.7 Assurance approach). For instance because threats, policies, assumptions and objectives are rather formal, we can ensure that all objectives are there to cover existing threats or policies, that our assumptions doesn't prevent the implementation of our security feature (see 3.7.1 PP and ST evaluation and to see an example of coverage test, see 3.5.2c. Threats).

In short common criteria bring some structure to security related document (Assessment review, specification, testing, audit document,) and some guideline in the process of getting some assurance about our product.

The rest of this section will give detailed information about every part of a Common Criteria based document and some information about the process involved.

3.2 History

In the first part of this document, we get to the conclusion that security has some importance in the development of a product (from the developer's and from the consumer's point of view). Of course this obvious conclusion is shared by lot of people. Since the early 70s, security become a concern, therefore lot of countries were trying to create some "criteria" regarding security, and that's why we are focusing on common criteria, in short: The Common Criteria speficifications are is the result of many criteria developed around the world.



In the early 1980's the Trusted Computer System Evaluation Criteria (TCSEC) was developed in the United States, in Europe (mostly France, Germany, the Netherlands and the United Kingdom), the Information Technology Security Evaluation Criteria (ITSEC) version 1.2 was published in 1991. In Canada, it was the Canadian Trusted Computer Product Evaluation Criteria (CTCEC) (version 3.0 was published in 1993) and it was already an attempt to unify criteria from Europe and United States (ITSEC and TCSEC). In the meantime, the Federal Criteria for Information Technology Security (FC) version 1.0 was published as a second approach to combine North American and European concepts for evaluation criteria.

As already said before, lots of people were trying to get some "criteria" in order to evaluate the security of their product (or the security of products they intended to buy). That's why in 1990, the ISO organization begin his work to develop a standard evaluation criteria for general use.

In June 1993, the sponsoring organizations of the CTCPEC, FC, TCSEC and ITSEC began a joint activity to align their separate criteria into a single set of IT security criteria that could be widely used.

In January 1996, the common criteria editorial board published version 1.0 of the common criteria. This document was approved by ISO in April 1996 for distribution as a draft.

After more than a year of review, feedback, revision and some reorganization in the common criteria organization, the 2.0 "beta" was published. Now we use Common Criteria version 2.0.

For historical and consistency reason, we still use the term "common criteria" to refer to what's officially called "Evaluation Criteria for Information Technology Security".

3.3 Scope

The common criteria weren't developed to fit the specific needs of one intervenient of a software development, it tries to cover in an independent way, all the aspect from the consumers to the evaluators, we will describe here what users could expect to get from common criteria.

3.3.1 Users

Common Criteria defines a way to present security related information in a methodical, formal and clean way. That's the main interest for users; they get something they can use to compare a product with another.

The different level of abstraction provided by common criteria also allows some clients to inform a software vendor about their security needs. For instance one could want the product to fit a given protection profile, to enforce a given level of security (EAL), etc. (see later section to get a description of those concepts).

Note: please note that Common Criteria identifies users as consumers.

3.3.2 Developers

Developers (and software vendor) can get many things from common criteria:

1. A CC construct that will allow them to make claims about their product.
2. A list of threats that can affect the system, objectives to counter those threats and a list of functional requirement to achieve those objectives. Basically they get some pieces of information to specify their system as a secure system.
3. Some references and sample documents for other products in order to help them address security issues for their own products.

3.3.3 Evaluators

One of the main objectives of the common criteria is to allow the evaluation of a product; security criteria are expressed in a standard way. This allows comparison between various products, and to test the conformance of a product with external security requirements.

Common criteria also provide a set of general actions that an evaluator could follow to get an idea of the product from a security point of view.

CC also provide a "measure" of security, in the form of an assurance level (see 3.7.2 Evaluation Assurance Level (EAL) and other assurance related classes), this measure relevant information for fast evaluation of a product.

3.3.4 Others

Other people can find some interests in common criteria. For instance:

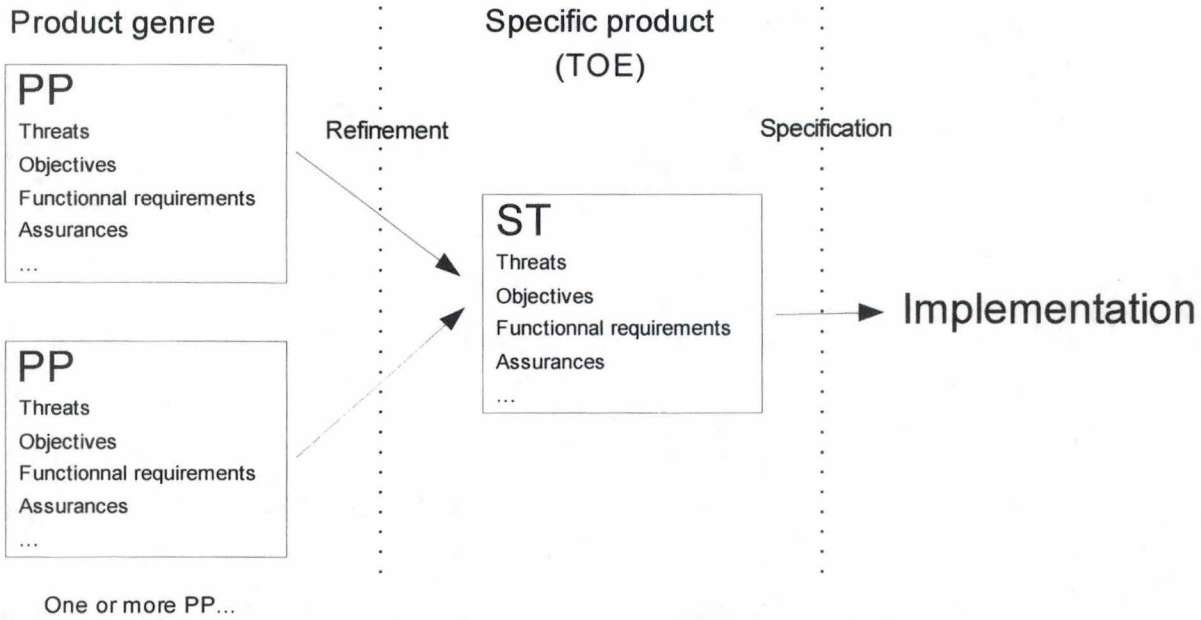
- Auditors, both internal and external, common criteria will help them to make assessment about the adequacy of the security of a system.
- Security architects and designers responsible for the specification of the security content of IT systems and product.
- Accreditors, they will follow the same path as evaluators in order to accept the use of a system in a particular environment.
- ...

In short: all the people interested in some way by the security of a product can get some useful information from a document based on the common criteria. A strange effect of that is that system cracker may also find useful information. That could lead us to another: Is it always worth to reveal information about the security of a product? This is another large topic related to security; it depends of lot of criteria's (who use the system? who may want to affect it? what's the real threats related to this system? etc...). It's also the ground for some debates about open source versus closed source projects. It's a huge discussion that involve development methodology, user consideration, philosophy, therefore it's out of the scope of this document. We will start from the point of view that it's up to each company to decide if they publish or not document about security, but no matter if the make it widely available or not, it's still important to get those information in a structured form. That's what of the objective of the common criteria.

3.4 Overview

We know that common criteria are like "the union of all the other" criteria. We've also seen that it's not "just a developer stuff" and that it can be useful for lot of people. Now, it's time to explain what exactly common criteria are and how they try to achieve their goals.

Common criteria define multiple level requirements, from the most generic up to the application. The construction is made from an abstract list of requirement, called "Protection profile". From this list, author will derive a list of requirements suitable for their application; this list is called Security Target (ST). This is a refined requirement list that begins to look like "requirements". The final step will be to compare those requirements with the TOE (target of evaluation, in other words: our system).



The next paragraph will focus on better definition of the PP, ST and TOE.

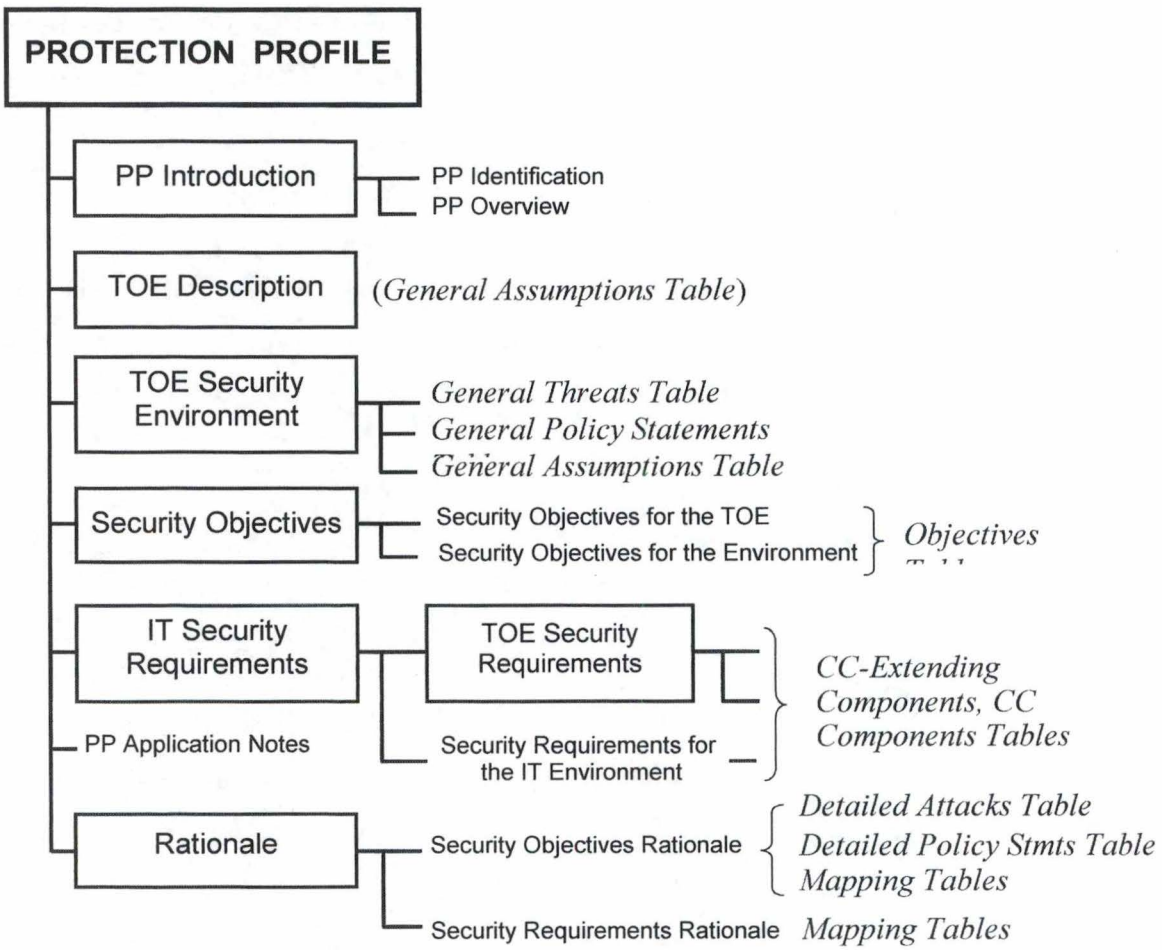
3.5 Protection Profile (PP)

It's hard to define protection profile more than "a list of threats, objectives and functional requirements covering the security of a specific domain". Because it's often easier to get into a concept with some examples, we will use a sample during our explanation. As an example we will use information extracted from the protection profile that exists for operating systems[15]. At the end of this section, there will also be an example PP related to outputware.

[15] See "Windows 2000 Security Target ST Version 2.0 18 October 2002".

The main interest of protection profile is that they're rather structured. Therefore they offer a common and clean representation of security threats, objectives and functional requirements. The formalism used allows performing some "integrity test" on the protection profile to ensure that it is coherent and complete.

The next figure shows the content of protection, and next sections will provide more information about each of those section.



3.5.1 Introduction and system overview

The first part describes the TOE, in other word, the kind of product for which the PP is suitable. It contains a description of the system. It's the first necessary step to limit the scope of the protection profile.

After reading the introduction, anyone should be able to determine:

- If the protection profile applies to the product he is concerning about.

- If the protection profile applies to his company (some protection profile are targeted to specific environment).

3.5.2 Description of the security environment and TOE description

An environment can be described in many ways, CC chooses the "threats approach", and secure environment is described by "what can impact the system?". Of course, a system isn't standalone stuff. It's always part of something. So external rules need to be applied to discover the system and analyze existing threats. Therefore the description will contain 3 things: assumptions, polices and threats. We will detail each of those parts in the next sections.

a. Assumptions

Assumptions are simple assertions regarding the system. This first step is helpful to reduce the scope of the PP. Because 100% secure software can't be reached, it's good to specify what part of those percent won't be reached anyway.

Examples of assumption are:

Name	Assumption	Discussion
A.COTS	The TOE is constructed from near-term achievable, commercial off the shelf (COTS) information technology.	This assumption is a key driver in determining the nature of the expectations toward, and hence the requirements to placed upon, the TOE.
A.MALICIOUS-INSIDER	The TOE is not expected to be able to sufficiently mitigate the risks resulting from malicious abuse of authorized privileges.	It is not reasonable to expect near term COTS products to provide sufficient protection against the malicious actions of authorized individuals.

As you can see, assumptions really limit the scope of the security achieved by the TOE. It's important to achieve the idea of completeness of a PP. A PP is complete... if it describes correctly what it can't achieve. It also "put some reality" in the concept: PP aren't magical solution to get a system secure, it's a real solution to get the most secure system actually realizable with current technologies.

b. Policies

Policies are general rules applied in the system. They can be seen as tools to make the TOE secure. Examples of policy are:

Name	**Policy**	**Discussion**
P.ACCESS	Access rights to specific data objects are determined by object attributes assigned to that object, user identity, user attributes, and environmental conditions as defined by the security policy.	CSPP-OS supports organizational policies that grant or deny access to objects using rules driven by attributes of the user (such as user identity, group, etc.), attributes of the object (such as permission bits), type of access (such as read or write), and environmental conditions (such as time of-day).
P.ACCOUNT	Users must be held accountable for security-relevant actions.	CSPP-OS supports organizational policies requiring that users are held accountable for their actions, facilitating after-the-fact investigations and providing some deterrence to improper actions.
P.COMPLY	The implementation and	The organization will meet

	use of the organization's IT systems must comply with all applicable laws, regulations, and contractual agreements imposed on the organization.	all requirements imposed upon it from the outside; for example: Government regulations, national and local laws, and contractual agreements.
P.DUE-CARE	The organization's IT systems must be implemented and operated in a manner that represents due care and diligence with respect to risks to the organization.	It is important that the level of security afforded the IT system be in accordance with what is generally considered adequate within the business or government sector in which the organization is placed.

The policy are the first kind of "must implement" for a secure application. Security as some impact on privacy ("can the application store any information about his users?"), law ("What kind of cryptography can the application use?"), internal company policy ("Access to this resource need to be restricted, even if it's not unsafe for the application").

c. Threats

Examples of threats are:

T.AUDIT-CONFIDENTIALITY-TOE: Records of security events under control of the TOE may be disclosed to unauthorized individuals or processes.
TOE security depends in part on the ability of the TOE to detect and report the occurrence of security relevant events, to determine the identity of those responsible for such events, and to protect the event records from unauthorized access, modification, or destruction.
T.AUDIT-CORRUPTED-TOE: Records of security events under control of the TOE may be subjected to unauthorized modification or destruction.
T.CRASH-TOE: The secure state of the TOE could be compromised in the event of a system crash.

For the TOE to protect the information it controls, it must remain in a secure state, including after recovery from a system failure or discontinuity of service. System crash can occur with inadequate mechanisms for secure recovery. Data objects and audit information may be modified or lost and system software may be corrupted.

As you can see, it remains large and could cover any kind of operating system (in fact those samples could fit non-OS product too). That's an important point about PP: they don't care about technical details. That's important for two things: The first is that it can be used and reused for any OS (so it could be used to compare OS), the second is that this document is understandable (up to a certain limit) by non-programmer, so PP can have some interesting for consumer, third party, …

3.5.3 Security objectives: how the TOE will address the threat.

We now have a list of threats and policies that need to be addressed in our system. From that list, we will create a list of objectives. Assumptions will help us make "realist" objectives.

Examples of objectives are:

TOE Security Objective	**Corresponding Threat or Policy**
O.ACCESS-TOE: The TOE must provide public access and access by authenticated users to those TOE resources and actions for which they have been authorized. This will be accomplished with high effectiveness.	P.ACCESS
O.ACCOUNT-TOE: The TOE must ensure, for actions under its control or knowledge, that all TOE users can subsequently be held accountable for their security relevant actions. This will be done with moderate effectiveness, in that it is	P.ACCOUNT T.TRACEABLE-TOE T.RECORD-EVENT-TOE T.AUDIT-CORRUPTED-TOE

anticipated that individual accountability might not be achieved for some actions.	T.AUDIT-CONFIDENTIALITYTOE
O.AUTHORIZE-TOE: The TOE must provide the ability to specify and manage user and system process access rights to individual processing resources and data elements under its control, supporting the organization's security policy for access control. This will be accomplished with high effectiveness. NOTE: This includes initializing, specifying and managing (1) object security attributes, (2) active entity identity and security attributes, and (3) security relevant environmental conditions.	P.ACCESS

From this example, you can see that those objectives are really based on the threat and policies. Because the purpose of this is to cover all threats and policiesies by an objective, we need to ensure that every threat/policy is addressed by an objective. It's like reading this table in the other orientation. Instead of having policy/threat related to objective, we need a list of objective related depending on threat/policy. Please note that this task could be easily automated, that's why a tool was developed to help in those tasks. (5.2 See SecGen – A helper to generate PP).

For our example, the table would look like:

P.ACCESS	**O.AUTHORIZE-TOE, O.ACCESS-TOE**
P.ACCOUNT	**O.ACCOUNT-TOE**

T.TRACEABLE-TOE	**O.ACCOUNT-TOE**
T.RECORD-EVENT-TOE	**O.ACCOUNT-TOE**
T.AUDIT-CONFIDENTIALITYTOE	**O.ACCOUNT-TOE**
T.AUDIT-CORRUPTED-TOE	**O.ACCOUNT-TOE**
T.RECORD-EVENT-TOE	**O.ACCOUNT-TOE**

Based on this simple sample, that's not enough, but if we look at our thread list, we have the threat T-CRASH-TOE. If we consider this thread we will get:

T-CRASH-TOE	NOTHING!!

And that's were it's interesting; we get a fast view of uncovered threat.

We now have some information about how to create objective list, and how to ensure they're complete and coherent. We can also see from where problems may come: if the threat or policy list isn't done correctly, everything can get wrong. Even worst, as we will see later, functional requirement will be wrong too. That's why threats/policiescies are a very important part of the common criteria. A tool can help, but it won't make things correct.

Please also note that outside that list, PP should contain rationale to explain the choices of the threats, policies and objectives. This will help reader understand why these threats exist, and if it's relevant for them.

3.5.4 Functional requirement

From those objectives, PP writer will build a list of functional requirement. Of course CC doesn't let us alone in this exercise, as we saw, CC define a list of class, subdivided in family, containing functional requirement. So the first step of a PP writer is to make his choice in all those possible requirements, determine how they address the objective, and then test if all the objectives are well covered. That explains how important are the objective in this construction.

Examples of functional requirements:

Functional Component	Name	Dependency for	Required to help address
FAU_GEN.1-CSPP	Audit data Generation	FAU_GEN.2 FAU_SAR.1 FAU_SEL.1-CSPP FAU_STG.1-NIAP-0423	O.ACCOUNT-TOE O.RECOVER-TOE O.RECOVER-SYSTEM O.DETECT-TOE O.DETECT O.OPERATE O.MANAGE O.DUE-CARE
FAU_GEN.2	User Identity Generation		O.ACCOUNT-TOE
FAU_SAR.1	Audit Review	FAU_SAR.2 FAU_SAR.3	
FAU_SAR.2	Restricted Audit Review		O.BYPASS-TOE
FAU_SAR.3	Selectable Audit Review		O.ACCOUNT-TOE O.RECOVER-TOE O.RECOVER-SYSTEM O.DETECT-TOE O.DETECT O.DUE-CARE O.OPERATE O.MANAGE O.COMPLY

From this table some questions may come:

The first one is the meaning of FAU_SAR.1, as said previously our functional requirement are divided in classes, family and components. CC defines a notation for a component, to allow unique identification:

FAU	The class.
FAU_**SAR**	The family.
FAU_SAR.**1**	The component.

CC defines a rather complete list of class and family in order to help PP author, this full list can be found in the part 2 of the Common Criteria document. Again, software could help us here: we're using rather structured informationinformation; a tool could present us this information in a clean way, so PP author will just "have to make their pick".

For instance, FAU_SAR.1 is defined as: "audit review provides the capability to read information from the audit record". In clear: the program must provide a way to make the audit information visible to someone.

The second question is about dependencies; dependencies are there to inform reader that some dependencies can't be achieved if others aren't. In this sample: Restricted audit review can be achieved if audit review isn't possible.

From this construct, we get:

1. A list of functional requirements for our application domain.
2. A way to prove they match our objective.
3. A way to prove our objective matches our threats and policies.

Please note that "prove" in this context doesn't mean that our application will be secure. We.W can prove our data are coherent, but we still need a correct list of threats and correct objectives related to those threats. That lead us to the conclusion that CC are a great help to make our app secure, but like any other kind of specification, they need lot of work to be correct.

3.5.5 Rationale

All this construction seems to be build from almost nothing. We'veW a coherent structure, but nothing to argue about his content. That's why protection profile contains a rationale section. In this section, PP author will have to argue on their choice, so the PP reader will be able to understand why this threat exist, why it's covered by this specific objective and how the objective is implemented by those functional requirements.

However because name and details about each policies, threats and objectives are quite understandable, it's often not needed to rationale all of the choice made during the auythoring of the PP.

3.6 Security Target (ST)

PP and ST are pretty close; the main difference is that ST focuses on one specific TOE (system, subsystem, software) where PP tries to cover more. ST and PP are structured in the same way, but aren't really authored in the same way.

ST is build from one or more PP, so the process is a bit different, you take one or more PP, choose what's relevant for your product, and you'll get your ST.

The resulting document is almost the same, the only difference is that the PP targets a range of product (a domain) and the ST addresses a specific product. The section most likely to change will be the rationale section: we will have more elements to argue on our choice, so the ST can be more specific and precise.

3.7 Assurance approach

Until now, we've focused on a list of possible threats, a list of objectives that need to be reached against those threats and in the end a list of functional requirements to achieve those objectives. It would be naïve to think that it would be enough to avoid any attack against a system, but it can give us some assurance about our system. This part will focus on how to develop and test our system to get some assurance about it.

Common criteria proposes evaluation in order to get some information about our system security. security. From the CC document, we get:

Evaluation has been the traditional means of gaining assurance, and is the basis of the CC approach. Evaluation techniques can include, but are not limited to:

a) Analysis and checking of process(es) and procedure(s);

b) Checking that process(es) and procedure(s) are being applied;

c) Analysis of the correspondence between TOE design representations;

d) Analysis of the TOE design representation against the requirements;

e) Verification of proofs;

f) Analysis of guidance documents;

g) Analysis of functional tests developed and the results provided;

h) Independent functional testing;

i) Analysis for vulnerabilities (including flaw hypothesis);

j) Penetration testing.

The CC philosophy (and the obvious one, generally accepted) asserts that the more effort you put in testing, the more your assurance about your system will be accurate. The effort is a vague notion; it can be greater or lower depending on the following criteria:

- **Scope**: what part of the system will you take in consideration?
- **Depth**: how far you'll dig into your system?
- **Rigor**: what kind of methodology you'll apply, how structural will your testing be done?

As it was needed for functional requirement, some structure is needed to make the evaluation of our TOE security. Common criteria use the same organization for assurance related things than those used for functional requirements: In other words: class, family and component.

In the next sections of this document, we will spend some time on some of the "assurance related classes".

Assurance Class	Assurance Family	Abbreviated Name
Class ACM: Configuration management	CM automation	ACM_AUT
	CM capabilities	ACM_CAP
	CM scope	ACM_SCP
Class ADO: Delivery and operation	Delivery	ADO_DEL
	Installation, generation and start-up	ADO_IGS
Class ADV: Development	Functional specification	ADV_FSP
	High-level design	ADV_HLD
	Implementation representation	ADV_IMP
	TSF internals	ADV_INT
	Low-level design	ADV_LLD
	Representation correspondence	ADV_RCR
	Security policy modeling	ADV_SPM
Class AGD: Guidance documents	Administrator guidance	AGD_ADM
	User guidance	AGD_USR
Class ALC: Life cycle support	Development security	ALC_DVS
	Flaw remediation	ALC_FLR
	Life cycle definition	ALC_LCD
	Tools and techniques	ALC_TAT
Class ATE: Tests	Coverage	ATE_COV
	Depth	ATE_DPT
	Functional tests	ATE_FUN
	Independent testing	ATE_IND
Class AVA: Vulnerability assessment	Covert channel analysis	AVA_CCA
	Misuse	AVA_MSU
	Strength of TOE security functions	AVA_SOF
	Vulnerability analysis	AVA_VLA

3.7.1 PP and ST evaluation

The first step is to make sure the things we want to evaluate are coherent with themselves, so we have to test our PP and ST specification with themselves. This part was partially considered in the section Security objectives: how the TOE will address the threat..

To evaluate PP and ST, Common Criteria proposes 2 classes, one for the PP (called APE for "assurance profile evaluation") and for the ST (called ASE for "assurance security target evaluation"). Because PP and ST are really close, with a different scope, the two classes are almost the same.

a. Assurance profile evaluation (APE)

Class	Family	Abbreviated Name
Class APE: Protection Profile evaluation	Protection Profile, TOE description	APE_DES
	Protection Profile, Security environment	APE_ENV
	Protection Profile, PP introduction	APE_INT
	Protection Profile, Security objectives	APE_OBJ
	Protection Profile, IT security requirements	APE_REQ

Like any CC class, APE is divided into some components; the following table contains the list of those components[16].

The common criteria specification provide a full description of each of those family, the important point, is that a "correct" PP, must fulfil all those family, so it must have:

- A correct TOE type description (a PP about operating system has first to describe, "What's an operating system in this document").
- A coherent description of the environment.
- An introduction describing the PP to allow classifying, identifying it easily.
- The PP contains a list of objective, with their rationale, the objectives are coherent with thread and allow countering them.
- The PP must contain a list of requirement, coherent with the objective; they must fully cover the objectives.

In other words, a PP must have the structure described in Protection Profile (PP)

[16] This list is correct if the PP doesn't contain external requirement, if it has, a new family need to be take in consideration (APE_SRE).

b. ASE assurance security target evaluation

Class	Family	Abbreviated Name
Class ASE: Security Target evaluation	Security Target, TOE description	ASE_DES
	Security Target, Security environment	ASE_ENV
	Security Target, ST introduction	ASE_INT
	Security Target, Security objectives	ASE_OBJ
	Security Target, PP claims	ASE_PPC
	Security Target, IT security requirements	ASE_REQ
	Security Target, Explicitly stated IT security requirements	**ASE_SRE**
	Security Target, TOE summary specification	ASE_TSS

Most of the families for PP and ST are the same, so we will focus on the differences.

- ASE_PPC: The ST must be a correct instantiation of the PP. Most of the ST is based on one or more PP, so it's necessary to check if the ST gives the same assurance about security than the PP did. If not, either the ST must not claim to be based on this PP, or the ST need to be modified to instantiate the PP.

- ASE_TSS: The ST shall contain a definition of the TOE, and make relationship between functional requirement and TOE features. As said before, that's the "rationale" section that needs to take the specific product in consideration instead of a generic, theoretical product.

With those different steps, we get some assurance that our PP is correct. Therefore, we have a clean working basis for the rest of our evaluation.

3.7.2 Evaluation Assurance Level (EAL) and other assurance related classes

At this stage, we have got a list of functional requirement that match objectives and those objectives match threats. And we've some assurance that our threats/objectives/functional requirements are coherent.

To get some more assurance about our system, we can perform two kinds of operation:

- Process related: how to develop, how to maintain, how to deliver, etc…

- Test related: perform test on our TOE to ensure that it performs well, even when misused.

Those operations can be applied up to a given depth. This depth will determine how « good » our assurance investigation was, it will be translated in the EAL level (from 1 to 7).

Common Criteria defines 7 classes to cover the way development process, maintenance, test, documentation, product lifecycle, vulnerability assessment and product modification are done. Each of those classes is subdivided into various families and components. The following table contains the required components for each EAL level in each class. So an EAL can be seen as a bag containing various components from the available families. Some "standard" EAL are defined by common criteria and named EAL1 to EAL7. There is, of course, still a possibility to create a "custom EAL", in fact, that's commonly used when a product meets EAL3 requirement, but also some of EAL4 requirements (they are EAL3 compliant enriched by some EAL requirements).

Next section will contain a brief description of each assurance class.

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Class ACM: Configuration management	ACM_AUT				1	1	2	2
	ACM_CAP	1	2	3	4	4	5	5
	ACM_SCP			1	2	3	3	3
Class ADO: Delivery and operation	ADO_DEL		1	1	2	2	2	3
	ADO_IGS	1	1	1	1	1	1	1
Class ADV: Development	ADV_FSP	1	1	1	2	3	3	4
	ADV_HLD		1	2	2	3	4	5
	ADV_IMP				1	2	3	3
	ADV_INT					1	2	3
	ADV_LLD				1	1	2	2
	ADV_RCR	1	1	1	1	2	2	3
	ADV_SPM				1	3	3	3
Class AGD: Guidance documents	AGD_ADM	1	1	1	1	1	1	1
	AGD_USR	1	1	1	1	1	1	1
Class ALC: Life cycle support	ALC_DVS			1	1	1	2	2
	ALC_FLR							
	ALC_LCD				1	2	2	3
	ALC_TAT				1	2	3	3
Class ATE: Tests	ATE_COV		1	2	2	2	3	3
	ATE_DPT			1	1	2	2	3
	ATE_FUN		1	1	1	1	2	2
	ATE_IND	1	2	2	2	2	2	3
Class AVA: Vulnerability assessment	AVA_CCA					1	2	2
	AVA_MSU			1	2	2	3	3
	AVA_SOF		1	1	1	1	1	1
	AVA_VLA		1	1	2	3	4	4

a. Assurance configuration management (ACM)

Today, no software is made from scratch, all developers use tools. From the compiler/linker up to the source management system, the case tools, the packaging tools, any tools can be of some help in software development. Two problems can come from this fact:

- The tool we use can make our TOE less secure (a bug or feature of the compiler that will let potential cracker know more information about our software, like fixed memory address, etc.). In fact, we can't be sure our tools are safe[17]. In the real world, no development group can imagine write *all* the tools they'll use[18], therefore we've got to have some trust in our tools.

- Lot of modifications can be done to our TOE (generally the program sources), if we want our software to be safe; we need a way to control what happen to our TOE[19]. We need to be able to track all change that can affect the TOE generation process. CC defines a class to address this development process related issue: ACM – Assurance configuration management.

The ACM class defines many families that should help people ensure that their TOE is generated from « safe sources », that versioning is coherent and complete. This process can be partially or totally automated and that's this level of automation that determine the EAL level (the most automated is considered as the most secure).

b. Delivery and operation (ADO)

A system can be totally secure, if we don't ship the right version (or a complete version), nothing can be assured. Note that it's also true for other requirement. Therefore it's very important to have a formalized packaging and delivery method. The requirements needed to achieve this objective are described in the class ADO.

This class tries to cover all aspect of product delivery, from documentation to the prevention of modification after the product was considered complete.

---

[17] For more information about "tool trusting" see http://www.albion.com/security/intro-18.html#pgfId-447197 and http://www.acm.org/classics/sep95/.

[18] And even in this scenario, it would need lot of development, so it would imply some trusting between the teams.

[19] As an illustration of this fact: in November 2003 a security flaw was introduced in the Linux kernel sources. The "bug" was rather hard to discover, but it was discovered with the help of CVS tracking tools. Without such tracking tools, this modification could have lead to an important security hole.

c. Development (ADV)

Security can't be achieved without the implementation of the « target security function ». The class ADV focuses on this development. It's based on a top-down approach, starting from the functional requirement produced by the ST up to the implementation, with various intermediate level of specification:

- TOE summary specification
- Functional specification
- High-level design
- Low-level design
- Implementation representation

CC allows three specifications style: informal, semiformal and formal. The style will determine how ambiguous will our assurance about the TOE be. It will also determine the EAL level associated with our TOE.

CC defines those 3 styles as:

- Informal: written in natural language, the only real rules to follow are the spelling and grammar of the language used (that can be any language). Informal specification should also define the terms used in the specification according to the context.
- Semiformal: Semiformal specifications use a restricted language (a subset of the natural language, for instance semiformal specification may force the use of some sentence structure). A semiformal specification can also be partially diagrammatic (data-flow diagram, state transition diagram, etc…).

Formal: Formal specifications are written using a commonly accepted formal language (mathematical language), with some informal explanatory information. There need to be evidence that it is impossible to derive contradictions, and all rules supporting the notation need to be defined or referenced. In short: it's the safest specification, it can't be contradictory and it can be proven.

Determining how to create specifications is out of the scope of this document, and out of the scope of the CC itself. The important information is that we must be able to "trace down" our specification from the most generic (functional requirement) to the implementation, to ensure that our implementation really implements the security functional requirements. Like for specification, this correspondence can be done using one of the three styles and again, the most formal is considered as the most secure (but also the most complex and time consuming).

CC provides some families that formalize the concept of "having the specification" and "tracing down the specification" and once performed give some assurance about our TOE implementations. This part could also be applied for other domains (not security related), we can use this to prove our software implement the needed requirements, no matter if they're security related or not.

Those assurance requirements can have a great impact of development methodology. Generally a development methodology will include some assurance regarding the quality of the product from a "functionality" point of view. Lot of work can be done in this part to make the existing QA fit the security assurance level requirement.

d. Guidance document (AGD)

As we've seen earlier, lot of security attack are avoidable, they result from misconfiguration, improper use or not updated software. To help address those issue, a correct documentation is needed, both addressed to administrator (How to configure the system to be secure?) and to user (How to use the system in a secure way?).

The class AGD focus on both user and administrator documentation. This documentation should explain in a « human readable » form how to address security issues related to the system. The respect of this requirement ensures that the product comes with correct documentation.

Again this requirement should be take in consideration with the product documentation (not related to security), in order to make the process more efficient, they are chance that the product will contains a single documentation, so it's important to write the documentation with consideration for security concepts.

e. Life cycle support (ALC)

Because security is a critical domain, that has impact of lot of aspect of the product. It's important to take security in consideration from the beginning. It's important to perform some control on the TOE during its development.

A software product is almost never a finished product: bug fix, new feature may come after the product was released. Like for development process, it's important to consider security during those enhancements.

The class ALC focuses on control and discipline in the development process and maintenances activities.

Life cycle support isn't specific to security, the important point to remember here: think about security when modifying the software, when a patch will be released, etc.

f. Tests (ATE)

When asking "how to be sure the product works?" the easiest answer is "Test it!". For security, it's basically the same answer. We need to test our software if we want to give some assurance about its security. The ATE class requires the product to be tested (coverage, depth, functional).

Test, of course, doesn't apply only to security, so again it's important to take the current testing methodology in consideration before performing the test. Performing this requirement, like lot of other requirements, will often be an integration of security in the existing specification/QA/testing process of the development/maintenance.

It's even more important for testing, because lot of test will get some influence from the security functionality implemented in the TOE. For instance if we added user login to ensure security, some test will need the user to login. It has two consequences, the first: it's useless to add a specific "correct login test", it's implicit. The second one: the other test needs a user login to be done, so they're slightly modified. It seems obvious, but this simple example shows us that security testing cannot be made totally independently from the rest of the test.

An important point about testing (security testing and other testing) is that it's often better to make a third party (can be internal to the company or external, but not the feature developer) to perform the testing. For security related issues, some companies are specialized in security testing. It has two advantages:

- The developer, that's a human fact, does impact testing: we develop the software the way we thought it may be used, and we test it the same way. A third party tester will not be impacted by the development process and thus can provide better testing.

- From a marketing point of view, it's often better to be able to claims that an external company has tested our software. That's also one of the roles of common criteria: have some external references for security claims.

g. Vulnerability assessment (AVA)

Any system can be abused, no matter how we pay attention to security in our development; a system can still contain flawa. It cannot be avoided, but we can get an idea about the possible attack that our system may encounter. For a good security level it's also important to document those possible flaws in order to get a fair view of the system's security.

For instance a common vulnerability is the strength of the encryption used, how long will a brute force attack take to crack a password? How does it depend of the password length? That's a very important question, because it will give idea about how long data can be considered as safe. If an application uses encryption that can be cracked in 60 days, the users need to know that they can't use the system to safely store data that are valid longer than 60 days.

The CC propose to analyze the possible information flaw exploit (where does the hacker will « get into the system »), and like other assurance requirement, it can be done more or less formally, and it will determine the EAL level claimable by the TOE. Next step is to analyze misuses of the TOE. The main objective is to disallow an administrator to configure it in an insecure way making users believe it's secure. So developers must remove ambiguities in the software (see ADV) and documentation (see AGT).

The last step of this class is stress testing the security. Trying to abuse the system (internally, or externally[20])

[20] Some companies even go further, they propose a system on a network and ask pirate to get into it, with a prize for the first one who succeed. Of course it's more a marketing operation ("see how it's secure, 2 months and still not hacked!") than a vulnerability assessment operation, but it can still be helpful to get assurance about the product.

4 Example

We'll make a sample common criteria PP in order to get a more precise idea of what it can be. First of all, we will try to describe on what kind of product the PP will apply then we will expose the threats, objectives, policies and requirements associated with this domain.

4.1 Usage model

Océ, one of the leaders of the outputware market defines three main "usage models" for his output management software. We will use them as the basis for our sample PP. Because these "usage models" all concern the same kind of activity and are rather close to each other, we will only consider one PP, with some additional information highlighting the differences between the models.

4.1.1 Overview of usage model 1: Office printing

When people think about printing, generally the first idea that comes in mind is printing a document, they've realized with a word processor. Hitting the "print button", then they expect to get their document printed on some printer in their office.

That's the first usage model "office printing". In the Océ context, the main differences is that it doesn't print directly on the printer, but on a dedicated server, this dedicated server will perform some more task, like choosing the appropriate printer (depending on the current load of the printer, the document size and format, the capability, the available printers, their locations, etc.).

After that the user goes to the printer and takes his document. This simple fact leads us to an important point about security: the weakest link, our software can be as secure as possible, if the document is confidential, the biggest threat is physical: how could we avoid somebody to steal a document printed by another worker?

From this fact, we can derive some information about office printing security:

- The software can't efficiently address threats related to data stealing. The company needs to be able to control who has access to the printers and when, and it's outside the scope of this document. Some printers also provide functionality to force a user to be present and enter some password when he wants to print his document (thus printing becomes a two step process: print the document from his software then ask the printer to print his document).

- Data corruption/modification can and need to be addressed by software. We need to get assurance that the document printed is the document that the user wanted to print.

- The dedicated server should be able to identify who is printing, in order to have a control over the resources used by this user. This is mainly to avoid abuse of consumable resources.

4.1.2 Overview of usage model 2: Document production center

Another way people approach printing, is via "copy shop", and other dedicated business. People format their document in a suitable format (postscript, PDF,...) and comes to a specific place to get their document printed (it can be done via floppies, CDs, intranet, internet, ... any suitable transport medium). It often involves larger document, higher number of copies, and need faster printer. An operator is needed to configure the printing and control it.

This is the second usage model for Océ outputware, called Document Production Center (DPC).

The main differences with Office Printing are:

- Need of an operator.

- Larger document.

- Higher number of copies.

- Faster printing (100-200 page per minute)

From a security point of view, it has some differences:

- There are third parties (company's internal or external) involved in the printing, and again it can be the weakest link. More control is needed if the documents are confidential (specific contract, etc…), again it's not software related.

- Data corruption can occur in two places: from the document source to the operator and from the operator to the printer. This issue can be addressed (partially) by software if the document is transported via a network.

- An important aspect to consider is the nature of the document. Large printing are often dedicated to a larger audience, therefore they are often less "private" or "confidential" than document in the Office Printing usage model.

- User identification is also needed, but can be addressed differently if the transport medium implies human contact. (There is less need for secure user identification if he needs to come personally to the operator, but the current trends are of course to submit electronically via some network or Internet).

- The cost of security flaw can be higher: if a user abuses the system to get large amount of copies printed, it can involve bigger cost. (More consumable used). On the other side, the presence of the operator guarantees some permanent control over the system. ("Why is this printer printing 20.000 copies of the same document?").

4.1.3 Overview of usage model 3: Electronic data processing

The last approach to printed document is the one everybody knows, even people that never touch a computer: large printing performed by companies, government, etc… Everybody received taxes notification, assurance related document, or even advertising. These kinds of printing are generally made on the biggest printer.

This third usage model is called "Electronic data processing". It has some important differences with DPC:

- Even larger amount of printed document

- Each copy is often customized. (It's not more a single PDF submitted by an author).

- No implicit operator. (The document goes from the application to the EDP, with some transformation performed by filters, it's a largely automated process). Of course on such big system, a "production operator" is needed to control the printer, but he doesn't have to submit each copy to the printer, like in the DPC model.

- Printers are able to print more quickly.

From a security point of view it's the most complex model, first of all there is a client/server configuration that perform complex task, without the implicit presence of an operator. Document are customized, therefore often contains personal information (from a list of name address, up to the more personal information like medial result).

If we look at the existing threat more deeply:

- Data need to be secure during all the processes; no modification or even reading of data could be done during the process. On the other side this process will most likely stay local to the printing company, network access will stay local and print server won't be exposed to the whole intranet. It will help the software address all network related threat, because the possible attacker can be easily controlled.

- If someone corrupts the data it can be very time consuming (thus expensive) to re-achieve the operation. Unlike other domains that don't have material output, a system reboot or software operation can't be enough.

- Reliability is very important; if the system becomes unavailable it will almost always have great impact on the company work. "Peoples don't print for fun."

- Because lot of preprocessing is done by the outputware on the document, and following the simple rule of the "weakest link", all those filter need to be fully secure. Their flaws will have an impact on the whole process.

- Because of the important resource used in this kind of printing, correct client application identification need to be performed on the server in order to avoid printing from "unofficial" sources.

- Because of the great complexity of this kind of printing, it's reasonable to assume that administrator that are behind those system have a correct knowledge of the system. In theory, the software doesn't need to cover "beginners related issue", like automatic software updating, software update notification, etc…

4.1.4 Common overview

From those overviews, we can extract some resemblance from the three usage models:

- All the usage models address printing, that is obvious, but it has some impact on security, for instance the weakest link will often be once the document is outside the system.
- Every configuration as some server/client architecture, even the most simple.
- User differentiation is needed in all models in order to prevent resource abuse.
- Cryptography can be needed for privacy, even for office document printing (avoid "out of the office" data stealing).

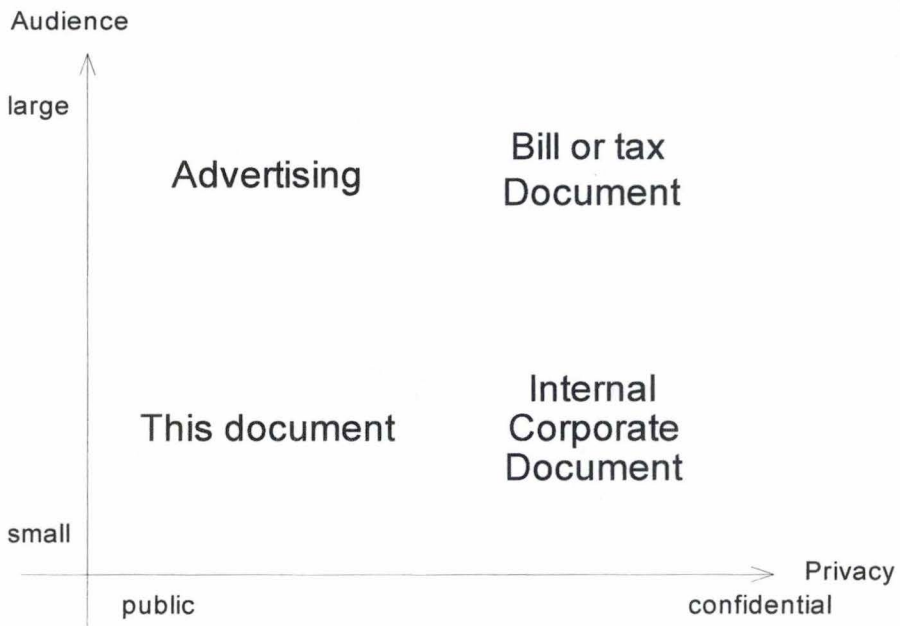
The main differences are on three levels:

- The customization applied on the document: in order the complexity of the filter. From a simple « change paper format » in Office Document Printing, up to the most complex « bill calculation algorithm » in EDP model.
- The amount of printed page.
- The length of the process
- The presence of an operator.

The similarity will allow us to get a « common PP » for our development. And the differences will force us to consider some differences in the policy and assumptions that we can apply to the system.

It's also very important to note that printing is nothing… without a document. The need for security in such applications are really determined by the importance of the documents, it will almost always be a big problem if documents get modified during printing, but it can be more or less important if the document can be consulted by an unauthorized third party (depending if the document contains personal data or not, etc…). In our development, we will consider « normal document », in other word the kind of document that we get everyday from public to confidential, but not specific or rare document like we could found in critical environment like nuclear power plant.

Based on the privacy and audience criteria, we can represent possible document in the following schema:



In our development, we will consider those 4 documents:

- Document like this one: predisposed to office printing. We could imagine print them in a document production centre. It's neither confidential neither targeted to large distribution.

- Internal corporate document, like product specification, bug reports, market study. Those documents are confidential but have a very small audience. That's also the kind of document that would apply for Document Right Management.

- Advertisement/newspaper: generally they're targeted to large audience, the document itself isn't confidential, but the filter applied on them can involve some personal information (name, postal address …). Generally one single instance of a filtered document will have no value, but if someone gets access to the whole address list, it's a more important threat.

- Bill or taxes related document: Those documents are largely distributed, but are confidential. A single instance of a document can be relevant from the view of a private. (« What's the salary of this guy who seems to be friend of my boss? »).

Based on all those pieces of information's it's not time to focus on our sample Protection Profile.

5 Protection Profile

5.1 Generating protection profile

Generating a protection profile from scratch is not an easy task, what kind of threats to consider, what objectives could be applied to map those threats, etc. The common criteria specification provides some help with a sample Protection Profile. Therefore we can start with a list of threats and objectives to cover a generic scenario. For our PP we will start from those criteria and modify them to fit our usage model. This is the usual way to create a new protection profile: build it from another one and because the Common Criteria committee provides one, we are using it.

Because the task of managing list is rather structured and easy to perform with appropriate software. Therefore a software was developed to help in the first draft generation.

5.2 SecGen – A helper to generate PP

SecGen was build to ease the task of writing the sample PP that comes with the thesis. Basically writing a PP consists of writing some list of threats, policies, objectives and functional requirements. The problem comes from the fact that all those items need to be coherent. That's the basic idea beyond this tool: helping author having coherence feedback on their PP.

5.2.1 Software features

Before thinking about the implementation we first have to clarify the minimum required feature for the tool. Because it's an internal tool developed for helping us in writing protection profile we are rather free on the choice of features. The conclusions of this reflection produced the following feature list:

- Edit list of threats, objectives, policies, assumptions and functional requirements using a modern windowed interface. In this context editing means: add items, remove items and modify items.

- Use an existing protection profile as the basis for a new one.

- Open format potentially usable by other software. This choice was made for interoperability with existing requirement software (for instance TestDirector de Mercury Interactive used by OSL).
- Run at least under windows operating system.
- Provide clear feedback about detectable incoherencies in protection profile and security target. Detectable incoherencies are:
    - Uncovered threats (no objective or assumption match the threat)
    - Uncovered policies (no objective or assumption match the policy)
    - Functional requirement that doesn't correspond to an objective (unneeded requirement)
    - Objectives that do not cover any threats or policy (unneeded objectives)
    - Assumptions that do not cover any threat or policy (unneeded objectives).
- The ability to generate report in a rich open format usable in common word processor (at least Microsoft® Word).
- Allow easy extension of the application to add new feature (Some possible extension are detailed in 5.2.5 Future development)

5.2.2 Technical choice

Based on this feature list, we decided to use some specific technologies to help us in our development.

The tools was developed using Microsoft .NET framework and the C# language. Micrsoft.NET was used because it allows correct performances on high end machine, easy debugging, and usable performance on low-end machine and low development time. The Visual Studio .NET IDE was used in order to ease the development. This language/platform choice was made against other possibility mainly for the following reasons:

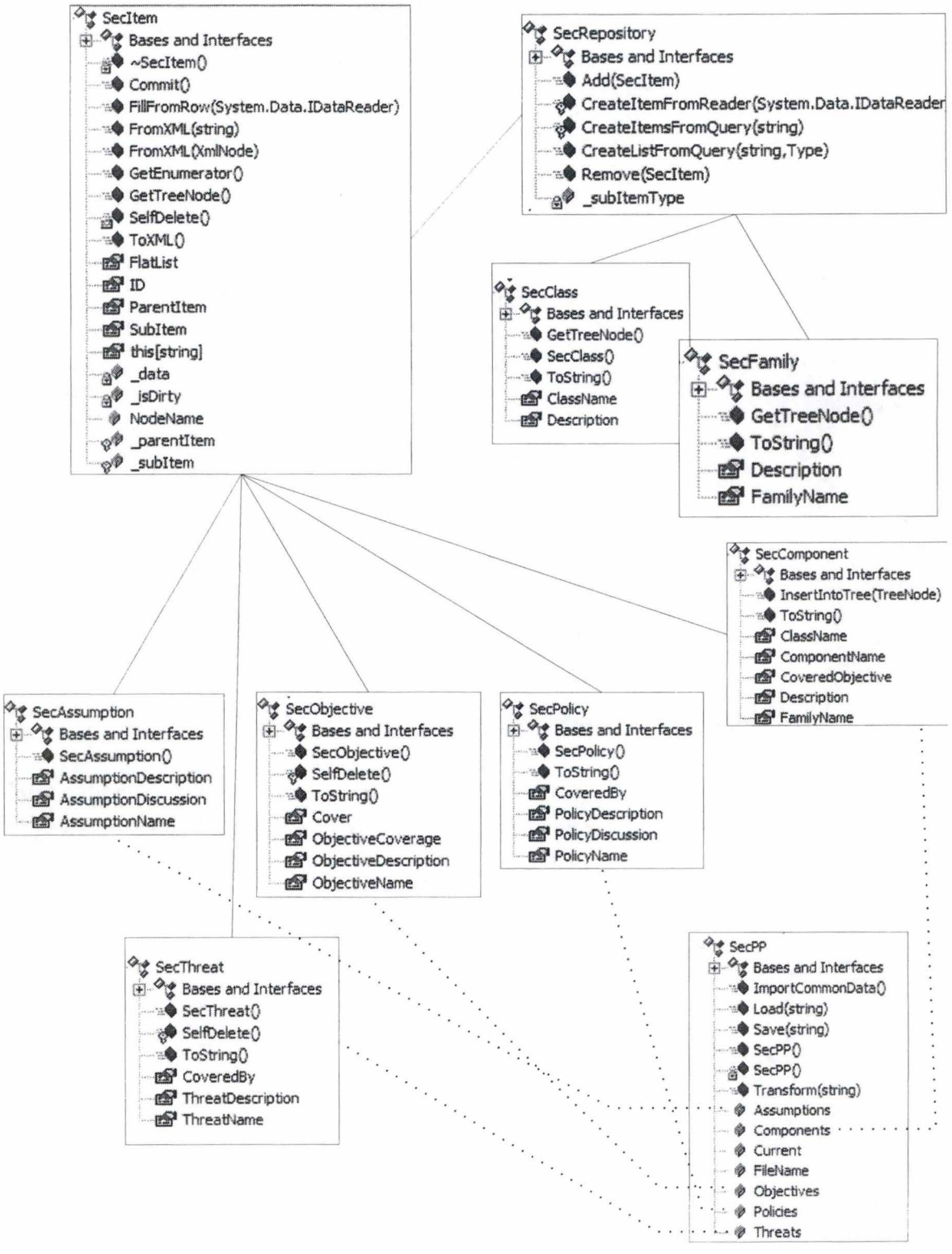
- Java was rejected because the development machine only contains 128Mb of RAM.
- Native languages like C++ are generally more error prone and require higher development time.
- Visual Basic was rejected because it doesn't offer correct performance in some scenario and doesn't allow a good OO design.

XML was used as the serialization format. This choice allows us to take profit of existing technologies to produce the output report in (X)HTML format and because it's one of the best format to allow future interoperability with other software.

The software was developed using a spiral development methodology: starting from a first version allowing with only a subset of the feature then adding more functionality. This choice was made because the software is heavily data-driven therefore most of the specifications of the software are inspired in the Common Criteria specification.

5.2.3 Software architecture

Based on the required feature and our technical choice, we are now able to think about the actual design of the application. One important fact is that data structure must be mapped to common criteria; therefore the design is heavily data-driven. The following schema represent the class diagram for non GUI related part of the software.


SecItem
Bases and Interfaces
~SecItem()
Commit()
FillFromRow(System.Data.IDataReader)
FromXML(string)
FromXML(XmlNode)
GetEnumerator()
GetTreeNode()
SelfDelete()
ToXML()
FlatList
ID
ParentItem
SubItem
this[string]
_data
_isDirty
NodeName
_parentItem
_subItem
SecRepository
Bases and Interfaces
Add(SecItem)
CreateItemFromReader(System.Data.IDataReader
CreateItemsFromQuery(string)
CreateListFromQuery(string,Type)
Remove(SecItem)
_subItemType
SecClass
Bases and Interfaces
GetTreeNode()
SecClass()
ToString()
ClassName
Description
SecFamily
Bases and Interfaces
GetTreeNode()
ToString()
Description
FamilyName
SecComponent
Bases and Interfaces
InsertIntoTree(TreeNode)
ToString()
ClassName
ComponentName
CoveredObjective
Description
FamilyName
SecAssumption
Bases and Interfaces
SecAssumption()
AssumptionDescription
AssumptionDiscussion
AssumptionName
SecObjective
Bases and Interfaces
SecObjective()
SelfDelete()
ToString()
Cover
ObjectiveCoverage
ObjectiveDescription
ObjectiveName
SecPolicy
Bases and Interfaces
SecPolicy()
ToString()
CoveredBy
PolicyDescription
PolicyDiscussion
PolicyName
SecThreat
Bases and Interfaces
SecThreat()
SelfDelete()
ToString()
CoveredBy
ThreatDescription
ThreatName
SecPP
Bases and Interfaces
ImportCommonData()
Load(string)
Save(string)
SecPP()
SecPP()
Transform(string)
Assumptions
Components
Current
FileName
Objectives
Policies
Threats

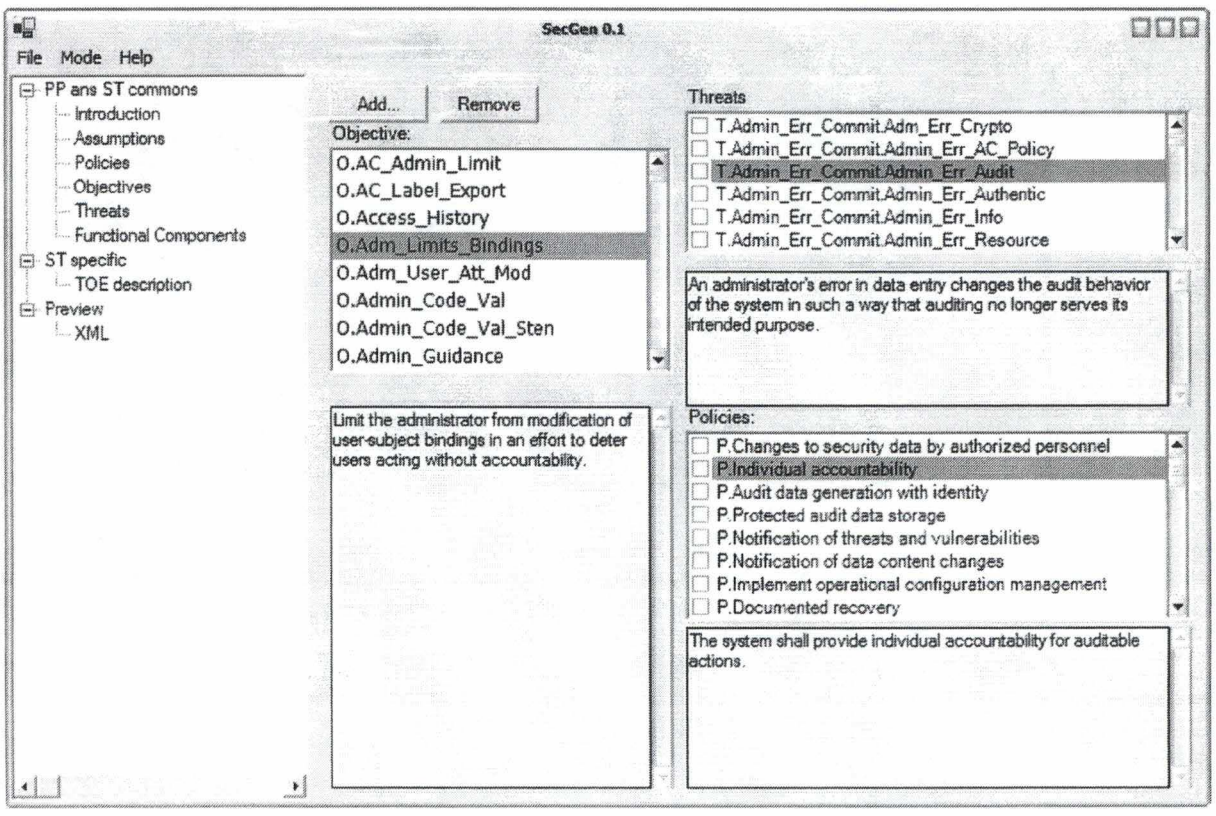
All the PP items class inherits from a SecItem class, this class is responsible for database and XML serialization. Each item has a unique ID. SecRepository is a special class allowing subitem, this class is needed to allow the hierarchy structure of functional requirement. SecPP is the global class that hold the current protection profile (or security target), it contains list of assumptions, functional requirements, objectives, policies and threats. A SecPP instance can serialize itself (includes his list of items) in XML format (see Annex D for a sample XML file produced by SecGen).

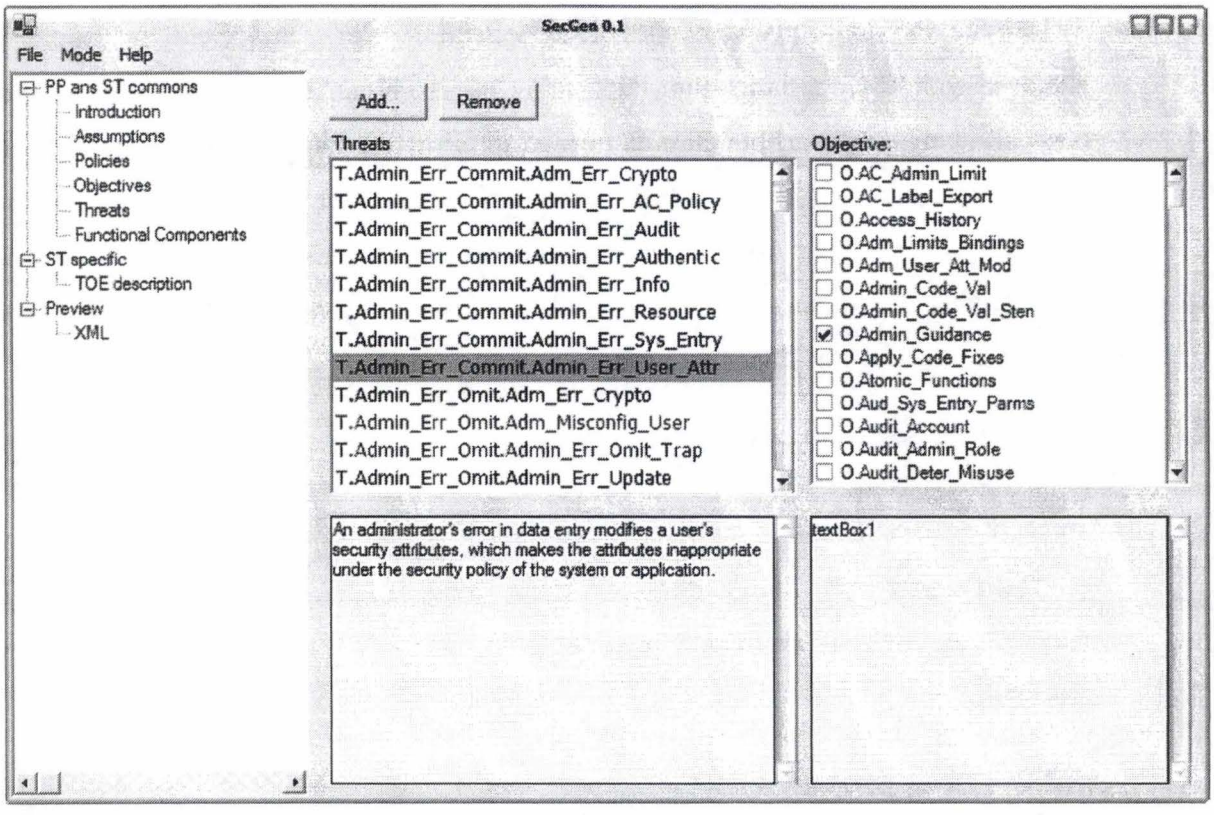
For GUI related part, it's a direct implementation using the limited RAD capability of Visual Studio.NET.

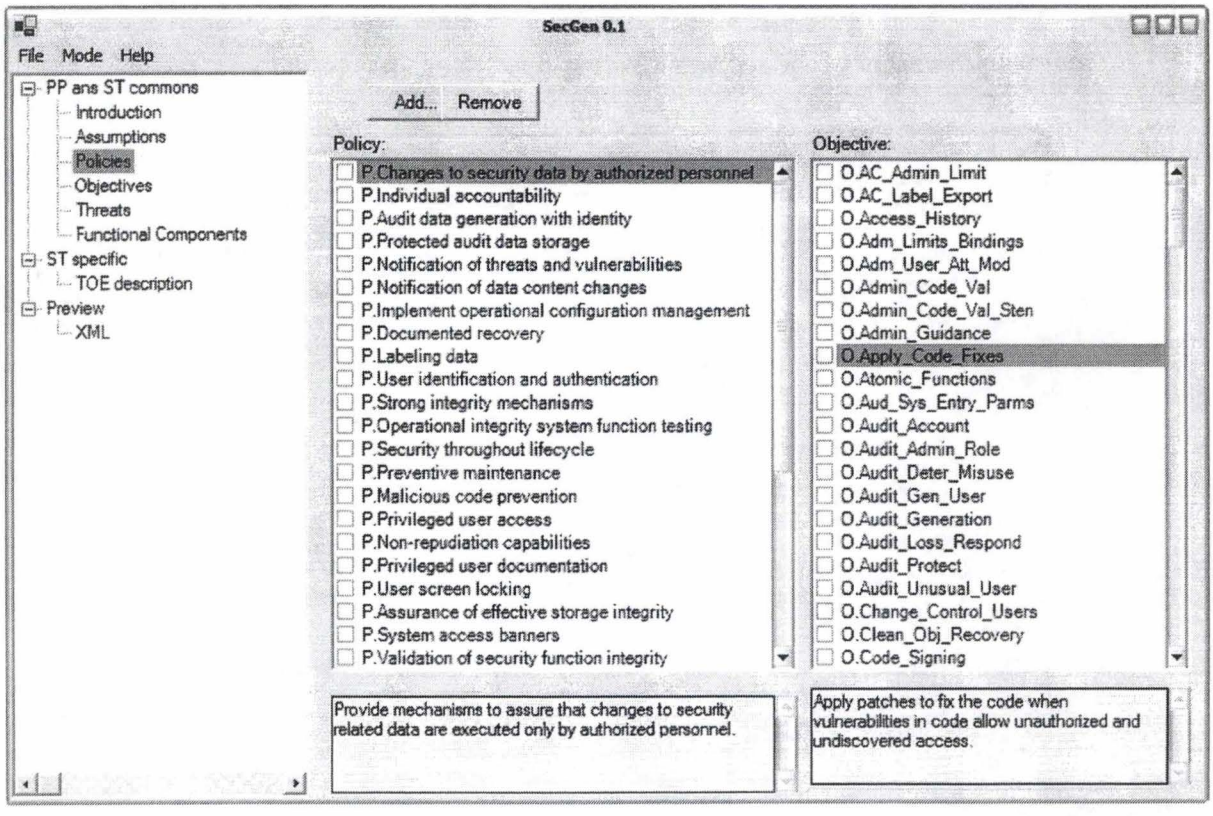
The Protection Profile generation is made using an XSL stylesheet (see Annex E), the output is an XHTML document importable in Microsoft® Word and various other word processor.

5.2.4 Tools presentation

Based on this design an implementation of tools was produced. Next figure represent various screenshot of the software editing objectives, threats and policies.




SecGen 0.1
File Mode Help
PP ans ST commons
Introduction
Assumptions
Policies
Objectives
Threats
Functional Components
ST specific
TOE description
Preview
XML
Add...
Remove
Threats
T.Admin_Err_Commit.Adm_Err_Crypto
T.Admin_Err_Commit.Admin_Err_AC_Policy
T.Admin_Err_Commit.Admin_Err_Audit
T.Admin_Err_Commit.Admin_Err_Authentic
T.Admin_Err_Commit.Admin_Err_Info
T.Admin_Err_Commit.Admin_Err_Resource
T.Admin_Err_Commit.Admin_Err_Sys_Entry
T.Admin_Err_Commit.Admin_Err_User_Attr
T.Admin_Err_Omit.Adm_Err_Crypto
T.Admin_Err_Omit.Adm_Misconfig_User
T.Admin_Err_Omit.Admin_Err_Omit_Trap
T.Admin_Err_Omit.Admin_Err_Update
Objective:
O.AC_Admin_Limit
O.AC_Label_Export
O.Access_History
O.Adm_Limits_Bindings
O.Adm_User_Att_Mod
O.Admin_Code_Val
O.Admin_Code_Val_Sten
O.Admin_Guidance
O.Apply_Code_Fixes
O.Atomic_Functions
O.Aud_Sys_Entry_Parms
O.Audit_Account
O.Audit_Admin_Role
O.Audit_Deter_Misuse
An administrator's error in data entry modifies a user's security attributes, which makes the attributes inappropriate under the security policy of the system or application.
textBox1


SecGen 0.1
File Mode Help
PP ans ST commons
Introduction
Assumptions
Policies
Objectives
Threats
Functional Components
ST specific
TOE description
Preview
XML
Add...
Remove
Policy:
P.Changes to security data by authorized personnel
P.Individual accountability
P.Audit data generation with identity
P.Protected audit data storage
P.Notification of threats and vulnerabilities
P.Notification of data content changes
P.Implement operational configuration management
P.Documented recovery
P.Labeling data
P.User identification and authentication
P.Strong integrity mechanisms
P.Operational integrity system function testing
P.Security throughout lifecycle
P.Preventive maintenance
P.Malicious code prevention
P.Privileged user access
P.Non-repudiation capabilities
P.Privileged user documentation
P.User screen locking
P.Assurance of effective storage integrity
P.System access banners
P.Validation of security function integrity
Objective:
O.AC_Admin_Limit
O.AC_Label_Export
O.Access_History
O.Adm_Limits_Bindings
O.Adm_User_Att_Mod
O.Admin_Code_Val
O.Admin_Code_Val_Sten
O.Admin_Guidance
O.Apply_Code_Fixes
O.Atomic_Functions
O.Aud_Sys_Entry_Parms
O.Audit_Account
O.Audit_Admin_Role
O.Audit_Deter_Misuse
O.Audit_Gen_User
O.Audit_Generation
O.Audit_Loss_Respond
O.Audit_Protect
O.Audit_Unusual_User
O.Change_Control_Users
O.Clean_Obj_Recovery
O.Code_Signing
Provide mechanisms to assure that changes to security related data are executed only by authorized personnel.
Apply patches to fix the code when vulnerabilities in code allow unauthorized and undiscovered access.

The user has the ability to add or remove items from those list, and select which one are relevant for his own protection profile or security target. Some items may appear in red (grey on printed version) because they have coherencies problems: for instance in the second screenshot two threats appear in red because they are uncovered threats (no objective match them).

5.2.5 Future development

The idea beyond this tool was to help the task of writing a specific PP for a specific document. But the tool was designed to be extensible. Future functionality could be:

- Global modification of a PP (merge, differences, etc…), because a clean XML document is used as serialization format, this is easy to implement.
- Integration with other tools, for instance "test case" could be generated for some functional requirements. The XML format used should allows use to make easy interoperability operation with third party tools.
- More authoring feature, for now the software only produce list of items. Some tools to add text around those lists could be helpful (add an introduction, conclusion, comments…).
- ST development: ST document have the same structure, but they are more TOE specific. That gives us the opportunity to add more functionality related to an actual development, for instance language and technology considerations.

5.3 Rationale for protection profile

Protection Profile is specified by the Common Criteria specification; therefore they are formatted according to their specific format (list of items). However because the purpose of this sample is to show the application of a subset of Common Criteria on a specific domain, more rationale will be given for the various choice, especially threats.

They are some important factor to take in consideration when reading the sample protection profile:

- It's a protection profile not a security target. The main consequence is that the protection profile needs to address a generic class of product and that they are no specific implementation to take care of. That also means that some assertion needs to be made on the future TOE. For instance: we will consider that the future TOE will use cryptography, that some kind of user authentification is needed, etc. (see Annex B. for a more complete description of those assertions). The advantage of this approach is that future ST author will be able to take the relevant part for their specific application without having to add lot of new threat or objective related to the technology they use. It's also a required step: without any assertion on the future TOE no threats could be extracted.

- Because Protection Profiles are fully described by Common Criteria, we choose to use the provided sample threat/objective/policies list. This choice was made for two reasons: first of all, the list was created by a specific committee of people with a great experience of security specification. The second is that various existing Protection Profile are written in the same way, this enforce many characteristic useful for anyone interested in PP: comparison with other Protection Profile, merging of many PP to make one ST, etc. On the other side, because Protection Profile vocabulary is generic, some explanations were added to describe why specific threats were choose and how they can exist in an "outputware" product.

- For Protection Profile threats, the purpose is to prove that they can exist in the future TOE, not if they are more or less likely to happen of it they can be totally avoided using a specific TOE (for instance if the TOE is dedicated to a single user, all the multi-user consideration will probably be unneeded).

5.4 The result

As exposed in 5.2 SecGen – A helper to generate PP, the first draft of the Protection Profile was generated using our tools. Then it was edited to fit our specific application type. The development methodology for the PP was a "spiral development": first draft was printed then analysed to discover missing threats or irrelevant threats. This review process was done many times in order to get a better protection profile.

In this context, "better" means that the protection profile will fit to the product domain. This phase is the most complex, we have to work on a generic inexistent product and we are not allowed to make to many claims about the future product. Lot of the decisions made about which threats, policies or assumptions to consider were made after some discussion with OSL employees.

Because a protection profile is a rather large document consisting of various lists, it's included as Annex C and not directly in this document. However, the experience gained by creating the protection profile is presented in this document.

5.5 CC evaluation

We have exposed common criteria; from this document we should now be able to understand how common criteria can be interesting (for coherences see 3.5.3 Security objectives: how the TOE will address the threat.). With the sample Protection Profile we have seen how Common Criteria list of threats, policies and objectives can fit in our sample applications. We will now explore some possible weakness or advantage of Common Criteria from a developer point of view.

5.5.1 A very good start

The first important point is that our protection profile contains lot of threats and polices with the corresponding objectives. It is one of the advantages of Common Criteria and the way they are build. Being able to take an existing Protection Profile and extend it to another product domain offer some guarantee that most of the possible threats are covered.

We were able to identify some threats that didn't seem applicable to our domain, sometimes about lot of reflection and talks. In some ways using the Common Criteria changed the way we needed to think about security threats. Some part of the reflection becomes "is this threat relevant?" instead of trying to list every possible threat.

5.5.2 Frontier between security feature and other feature

From the theory point of view, CC is coherent, logic and well documented. Starting from the assumption that nothing is perfect we should be able to ask some question about the relative independency of common criteria and the rest of the development. Some reader may have the feeling, based on this document, that security is seen like something apart, that's not the case; it's deeply linked to the rest of the development. That is one of the weakness of common criteria: it's nature force it to be independent from the rest of the project (it's about security not software development in general), but in reality the frontier between security feature and the rest of the application is a little more fuzzy (think about user management, is the login box a security feature or a simple part of the GUI? the code that make calculation based on the current user is part of security or not? What if data are first encrypted?).

5.5.3 Another source of specification

Another weakness is the way security target are build: they are derived from one or many protection profile, this seems logic and coherent but has one problem: The application specification are generally based on the client needs, that's the root of all the specification (even if someone the need are not expressed by the user directly but derived from his need), protection profile will add another root to the software specification. In other words: conflict between two choices of implementation will need more discussion to be resolved: will the user need be taken in consideration first or will the protection profile security requirement be considered. For instance: in our sample we have considered the use of cryptography, imagine that the user do not want any form of cryptography for some reasons (legal reasons for instance), this simple fact would invalidate lot of threats (and therefore objective, policies and functional requirement) in our PP.

5.5.4 Differentiation of PP items

Another weakness of Common Criteria is that it's only a list. That has the great advantage of being very structured and easy to read and build. On the other hand all the threats (and other PP items) are on the same level. It could have been interesting to be able to mark some threats as very important.

For instance in the sample developed in Annex B, all the threats were on the same level, even if some were more probable than other. For instance, the T.Hack_Crypto.Hack_Crypto_ChsnCy threat ("*The attacker discovers an unknown encryption key.*") is possible but very unlikely to happen with a modern cryptographic system. On the other side, threats like T.Dev_Flawed_Code.Dev_FC_Recovery ("*A system failure may alter the behaviour of the system's security functions in such a way that, upon recovery, it no longer properly enforces its security policy (TSP).*") are more likely to exist because they represent almost any possible bug in the software.

One possible improvement of Common Criteria would be to add different level of threats (and objectives) to qualify the relevance of those items. That information could be useful for real software development; it would set priorities on development and testing allowing the software to cover quickly the most important possible security failure.

It's important to note here that Common Criteria already offers EAL to make some distinction between important and less important security related functionality. But the distinction is mostly made on the way to get assurance about the security feature on a product not on the feature themselves.

5.5.5 Real interest of Common Criteria

We should have more time to experiment the common criteria: test them on a real application, derive a complete Security Target from our sample Protection Profile, and test it in a real world application to see what kind of impact the use of Common Criteria has. Unfortunately we don't have enough time for that.

Therefore questions about the real interest of common criteria remains open and will probably depends of those factors:

- Existing development model.
- Size of the application (like every specification, it's easier for smaller application).
- Time allowed for the project.
- Language and technologies used.

Some of those aspects are mentioned in section 0.Another very interesting aspect of common criteria are the assurance level provided by the use of those criteria. Being able to sell a product that fit in a given EAL will give the client some assurance about the product's security. It's relevant for the customers: they have more independent information about what they buy, it's relevant for the company that sells the software: they can prove their security claims.

Going further as future possible work. And that's were we are: we need more investigation to get an idea of the real impact of common criteria on a project, we have tried to show that common criteria are helpful during the development in a "perfect context", some work need to be done to determine what happen in real life.

5.5.6 PP/ST construction as a tree

Protection Profile and Security Target are all based on threats, policies and assumptions as the root, and then objectives and functional requirements are building from those 3 pillars. The problem with this approach is that there is no way using Common Criteria to test that the threats and policies are complete. A security target can get a good assurance level (see 3.7.2 Evaluation Assurance Level (EAL) and other assurance related classes) without being complete.

This problem is important of course, but also exists in any kind of specification: there is no way to efficiently test the coherence between client needs, applications correctness and actual development.

5.5.7 Useful existing list of items

Another fact that comes from the sample PP is that most of the sample threats, objectives and policies apply to our case. Of course they need some interpretations and adaptation (because we are in a different context), but most of them are applicable and relevant.

That seems rather logic: security feature are the same in lot of software, no matter what the software actually does cryptography technology, user authentification, audit management are the same (that also explain why there are so many development library dedicated to those domains: CryptLib, OpenSSL, Log4Net, Gina, ...).

5.5.8 Conclusion

What remains after all, is that common criteria are a mature specification, resulting of lot of work from various people around the world, used in certification process to get real assurance about product. The common criteria are coherent, extensible and offer a rather interesting framework when we need specification about security of a product.

Another very interesting aspect of common criteria are the assurance level provided by the use of those criteria. Being able to sell a product that fit in a given EAL will give the client some assurance about the product's security. It's relevant for the customers: they have more independent information about what they buy, it's relevant for the company that sells the software: they can prove their security claims.

6 Going further

Now, we should have some idea of what are Common Criteria and why it can be interesting for product development from a security point of view. Because security, even restricted to "pre-attack" stage, is a very large topic, lot of works can still be done. This work can be a starting point for various extensions, including the following points.

Technical approach

This document tried to be technology agnostic, of course, some assumptions need to be made, we are talking about computer, not something else, but no specific technology, language, platform was take in consideration. This fact can be an opportunity to dig more into specific technologies to see how they impact security development. For instance, based on document available on some platform (Java or .NET for instance) it may be possible that some of the functional requirement will be implicit, or will be implemented in a lot easier way. For instance, Microsoft .NET add the notion of trusted (or not) environment, to get some assurance about what an application can or can't do, buffer overflow impact are also different in such environment. Those concepts are not new but will probably change the way the software will be implemented, the impact of threat, objective and functional requirement during the development process.

That is even more true in real world were no application can be 100% secure, 100% bug free, developers have to make trade off between time and correctness of their application. A study about the existing technology from a security point of view could give very relevant information for the choice of a technology for the development of a new project. It would be also helpful after the development to get a more precise idea of the remaining threats surrounding the application (see 3.7.2g. Vulnerability assessment).

6.1 Development methodology

Authoring a common criteria related document is a task by itself, it should be integrated in the development process in a suitable manner and as already mentioned in Evaluation Assurance Level (EAL) and other assurance related classes (3.7.2), Common Criteria have some impact on other step of the development process (realizable feature, specification, testing, documentation, ...). It can be relevant to dig more in this topic, for instance by starting from one or more « real » development process and investigate about « how to hook CC in the existing process ».

It wasn't done in this document for two main reasons:

1. There are hundred of development methodologies, proper to each company, team, developer, focusing on only a few methods would probably lead us to specific consideration instead of a generic theory based approach.

2. Hooking CC into a development process need very good knowledge of a development process, that's not an easy task and it is very time consuming.

6.2 Improved tools

The tool developed with this document is mainly a « helper » for PP/ST author, it help author by providing « mostly used default », but doesn't hook in a specific development methodology or existing product documentation, product testing or product development. Lot of work can be done on this side, but again it's a very large domain, impacting security and lot of other development aspect.

6.3 Certification

We have considered CC mainly from a developer point of view, but common criteria are also widely used to give some assurance about a product. Therefore it can be interesting to dig more the certification process needed to get an EAL certification from a valuable third party. This work should be able to answer the following question:

- What steps are needed to get a certification?

- When does this need to be done? Can it be done after the product release?

- What are the cost and the benefits for such a certification? When does it become rentable?

- Are certifications a « fair view » of the product? How accurate is an evaluation?

6.4 Real impact on a project

Another point that can lead to interesting conclusion would be a work about « real benefit » from a security point of view of common criteria; does it make software more secure? The question is easy to ask but answering this question is a real challenge. The author of such a document would need access to detailed information about various products developed with or without common criteria in mind, information about user feedback, development time, etc... It could be also interesting to make « bench », for instance by asking to developer of approximately the same level to develop the same product one with common criteria and one without.

7 Conclusion

Based on this document, some conclusion can be taken about security in application in general, and the common criteria specification. It's important to note again that because security is a very huge subject, this document only cover a small subset of security issues in software applications.

First of all, we've explored the current market, two main ideas were exposed: security is needed, it's not fiction, real threats do exist, and we need to something against those threats. Second idea is that security doesn't seem to be the first concern of software vendor; they don't advertise that much on security, the retained explanation was that security is seen as an implicit feature. Based on this study, the first partial conclusion is that we need security, and we need to do it well. One way to enhance security in an application is to use a development methodology for security related part of the application.

The second part is the study of such a methodology. The methodology considered is the common criteria. The main reason of this choice is that it's an ISO standard and the result of many country specific security methodology. The sample Protection Profile help us to understand that Common Criteria is helpful to get a better view of security feature but also that it needs good integration in the rest of the development process.

Lot of works can still be done to get a more precise idea of security in today's software. This document put some of the first element: security is needed and there are some helpful methodologies to achieve more secure products, one of the is Common Criteria.

8 References

Common Criteria for Information Technology Security Evaluation, Version 2.1,August 1999.

CSPP - Guidance for COTS Security Protection Profiles, version 1.0, December 1999, Gary Stoneburner, U.S. DEPARTMENT OF COMMERCE, Technology Administration, National Institutes of Standards and Technology, Gaithersburg, MD 20899

COTS Security Protection Profile - Operating Systems (CSPP-OS) (Worked Example Applying Guidance of NISTIR-6462, CSPP), Version 1.0, Gary Stoneburner, U.S. DEPARTMENT OF COMMERCE, Technology Administration, National Institutes of Standards and Technology, Gaithersburg, MD 20899

Guide for the Security, Certification and Accreditation Of Federal Information Systems, Ron Ross and Marianne Swanson, June 2003

Windows 2000 Security Target ST Version 2.0 18 October 2002, Microsoft Corporation Corporate Headquarters One Microsoft Way Redmond, WA 98052-6399

Common Criteria, Security Evaluation, Part 1: Introduction and general model, August 1999, Version 2.1, CCIMB-99-031

Common Criteria, Security Evaluation, Part 2: Security functional requirements, August 1999, Version 2.1, CCIMB-99-032

Common Criteria, Security Evaluation Part 3: Security assurance requirements, August 1999, Version 2.1, CCIMB-99-033

CONTROLLED ACCESS PROTECTION PROFILE, Version 1.d, Information Systems Security Organization, National Security Agency, 9800 Savage Road, Fort George G. Meade, MD 20755-6000, 8 October 1999

Olivier Curet, Marc Mackinnon, *2003 Global Security Survey*, Deloitte Touche Tohmatsu, http://www.deloitte.com/dtt/cda/doc/content/Global%20Security%20Survey%202003.pdf, 12/03/2003

Operating System Structures to Support Secuirty and Reliable Software, Theodore A. Linden, National Bureau of Standards, 1976

Annexes

Annex **A**: table of OS flaws.

Annex **B**: A protection profile for outputware product.

Annex **C**: Common Criteria abbreviations

Annex **D**: Sample (minimalist) XML file for SecGen

Annex **E**: XSLT transormation file for generating Protection Profile Report from SecGen XML files

Annex A: Security issues in operating systems per year

OS	1997	1998	1999	2000	2001
AIX	21	38	10	15	6
BSD/OS	7	5	4	1	3
BeOS	0	0	0	5	1
Caldera	4	3	14	28	27
Connectiva	0	0	0	0	0
Debian	3	2	31	55	28
FreeBSD	5	2	17	36	17
HP-UX	9	5	11	26	16
IRIX	28	15	9	14	7
MacOS	0	1	5	1	4
MacOS X Server	0	0	1	0	0
Mandrake	0	0	2	46	36
NetBSD	2	4	10	20	9
Netware	1	0	4	3	1
OpenBSD	1	2	4	17	14
RedHat	6	10	47	95	54
SCO Unix	3	3	10	2	21
Slackware	4	8	11	11	10
Solaris	24	33	34	22	33
SuSE	0	1	23	31	21
TurboLinux	0	0	2	20	2
Unixware	2	3	14	4	9
Windows 3.1x/95/98	3	1	46	40	14
Windows NT/2000	10	8	78	97	42
http://www.securityfocus.com/vulns/stats.shtml					

Annex B: Protection profile for "outputware" product

The presentation of the TOE can be found in the main document. Because a Protection Profile covers a domains and not a specific product, we will focus on a generic "outputware product" that has the following characteristics (based on the presentation of the Océ usage model):

All the usages models address printing, that is obvious, but it has some impact on security, for instance the weakest link will often be once the document is outside the system.

Every configuration has some server/client architecture, even the most simple.

User differentiation is needed in all models in order to prevent resource abuse ("outputware" product need to know who print the document, even if the "who" is some group of users).

Cryptography can be needed for privacy, even for office document printing (avoid "out of the office" data stealing). Because it's one of the most efficient way to protect data, we can assert that an outputware will use some sort of cryptography.

Every usage model use the concept of privilege, some kind of user have administrator rights, some are simple user. In the most generic form: administrator and user could be a single entity in the application but they will always have different logical functions.

Outputware application need a good auditing system in order to keep track of the state of every document (print queue state for instance). This audit feature can be used for security purpose.

"Outputware" are real applications. They run on current hardware, with the known limitation of current hardware (possible flaw, speed, memory limitation, existing of overflows attack, etc.)

Those characteristics will be used as rationale for the various threats applicable to our TOE. Before the threats list we first to formalize our assumption about the future TOE.

1.1 Assumptions

Name	Description
A.Admin_Attitude	The administrator is considered as potentially hostile. He has good knowledge of the system functionality and the way to abuse the system. It can be a local or remote administrator. To consider the most generic case, we need to take in consideration porrly trained administrator to. The consequences will be that our threats will consider the admin as able to makes error and to abuse the system.
A.Eavesdrop_by_Out	Most of the time an "outputware" is separated from the rest of the network and not exposed to the rest of the world. We will consider this as the default: • TOE is not exposed to the internet. • Access is possible via the company's LAN.
A.Physical	The fact that we are talking about printed output force us to put a very strong assumption here: • The TOE can't do anything once the document is printed. The company is responsible for this part of the security of the process.
A.Password_Management	An assumption is made about password and other identification data. The TOE should provide correct guidance about the correct way to choose identification data to ensure TOE's protection. We consider that user won't expose the identification data to third party.
A.Admin	The TOE will run on existing operating system, the administrator is responsible for security issues related to the operating system and any other application running on the machines on which the TOE is executed. Admnisrtation include: virus checking, applying patch, configuring the operating system security to ensure that no other application can impact the TOE in an insecure way.
A.COTS	The TOE is constructed from near-term achievable, commercial off the shelf (COTS) information technology. In other word: it's actually realisable using current technologies.
A.Communications	Securing the communication from a material/physic point of view is out of the scope of the TOE security. Correct protection of wire, router, switch, access point must be guaranteed in order to ensure the secure state of the TOE.

1.2 Threats

Some threats are relative to misuses from the users, some are more related to some specific attack ("hacker's attack"). Rationale for those threats indicates why the threat is relevant with some specific information when needed. All the threats that fit in the "external hacked scenario" are specified with the [Attack] keyword in the rationale.

Name	Description	Covered by	Rationale
T.Admin_Err_Commit.Admin_Err_Crypto	An administrator misconfigures cryptographic functions or stores plaintext keys in insecure areas.	O.Audit_Account O.Crypto_Key_Man O.Crypto_Manage_Roles O.I&A_User_Action	4, 5
T.Admin_Err_Commit.Admin_Err_AC_Policy	An administrator's error in data entry changes the access control or information flow policy enforced by the system in such a way that it no longer serves its	O.Admin_Guidance O.Security_Attr_Mgt O.Security_Data_Mgt O.Security_Func_Mgt O.Security_Roles	5

	intended purpose.		
T.Admin_Err_Commit.Admin_Err_Audit	An administrator's error in data entry changes the audit behavior of the system in such a way that auditing no longer serves its intended purpose.	O.Admin_Guidance O.Audit_Admin_Role O.Audit_Loss_Respond O.Audit_Protect O.I&A_User O.Security_Data_Mgt O.Security_Roles	5, 6
T.Admin_Err_Commit.Admin_Err_Authentic	An administrator's error in data entry changes the authentication-enforcement mechanism of the system in such a way that it no longer serves its intended purpose.	O.Admin_Guidance O.Limit_Actions_Auth O.Security_Data_Mgt	5, 3
T.Admin_Err_Commit.Admin_Err_Info	An administrator's error in data entry makes system or application information unavailable.	O.Admin_Guidance O.Priority_Of_Service O.Resource_Quotas O.Security_Data_Mgt O.Security_Roles	5
T.Admin_Err_Commit.Admin_Err_Resource	An administrator's error in data entry makes system or application resources unavailable.	O.Admin_Guidance O.Security_Attr_Mgt	5
T.Admin_Err_Commit.Admin_Err_Sys_Entry	An administrator's error in data entry changes the intended entry policy of the system or application.	O.Admin_Guidance O.Security_Data_Mgt	5 (for instance allowing to print on unsupported paper format)
T.Admin_Err_Commit.Admin_Err_User_Attr	An administrator's error in data entry modifies a user's security attributes, which makes the attributes inappropriate under the security policy of the system or application.	O.Admin_Guidance O.Security_Attr_Mgt O.Security_Data_Mgt O.Security_Func_Mgt O.Security_Roles O.User_Attributes	5 (for instance: all the users becomes admininstrators).
T.Admin_Err_Omit.Adm_Misconfig_User	A change in the status of users duties do not get reflected in administratively controlled privileges and/or authorizations.	O.User_Auth_Management	5 (an user is able to do perform some action that he shouldn't be able to perform)
T.Admin_Err_Omit.Admin_Err_Omit_Trap	An administrator inadvertently leaves a back door port open after routine maintenance, allowing continuing unauthorized access by the service organization.	O.Maintenance_Access O.Maintenance_Recover O.Prvlg_IF_Status	5
T.Admin_Err_Omit.Admin_Err_Update	The organizational security policies changes but these changes are not reflected in all system configurations, resulting in circumvention and/or incorrect application of security policies.	O.Admin_Guidance O.Audit_Account O.Secure_Configuration	5 (imagine the following scenario: an employee get fired but he's still able to get access to printer queue)
T.Admin_Hostile_Modify.Adm_Hstl_Audit_Dstr	An administrator seeks to cover up misbehavior by destroying and/or falsifying audit data.	O.Audit_Admin_Role O.Audit_Protect O.I&A_User	5 (the simple fact the the administrator is human).
T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC	An administrator maliciously modifies access control attributes, allowing the administrator or other perpetrator to gain access and manipulative capability to	O.AC_Admin_Limit O.Audit_Account O.Audit_Loss_Respond O.Audit_Protect	5, 3

	organizational assets, contrary to organizational policy.		
T.Admin_Hostile_Modify.Adm_Hstl_Mod_IFC	The administrator maliciously alters information flow control policy to allow information to flow to inappropriate locations for unauthorized users access or modification.	O.Audit_Admin_Role O.Audit_Protect O.I&A_User O.Info_Flow_Ctrl_Admin	5 (for instance: an administrator misconfigure printer in a way that makes some user print in another office)
T.Admin_Hostile_Modify.Adm_Hstl_Mod_SEP	An administrator or user masquerading as an administrator maliciously modifies system entry parameters which would allow unauthorized access to an organization's protected assets.	O.AC_Admin_Limit O.Aud_Sys_Entry_Parms O.Audit_Admin_Role O.I&A_User	5,3 (this threats is part of the classic "pirate attack" as some were exposed in the first section of the main document) [Attack]
T.Admin_Hostile_Modify.Adm_Hstl_Mod_TSFCode	The administrator modifies the security-critical (TSF) code to weaken the security effectiveness of the TSF or introduce a new security breech.	O.Obj_Protection O.Sys_Self_Protection O.TSF_Mod_Limit	5,3 [Attack]
T.Admin_Hostile_Modify.Adm_Hstl_Mod_USB	The administrator modifies a user/subject binding which would allow a user to act on an object without creating an audit trail.	O.Adm_Limits_Bindings	5,3 [Attack]
T.Admin_Hostile_Modify.Adm_Hstl_Mod_UsrAttr	The administrator modifies or mishandles the users attributes or roles which allows users, unauthorized or authorized, to have the ability to perform inappropriate actions or could prevent a user from performing an authorized action.	O.Adm_User_Att_Mod	5,3 [Attack]
T.Admin_UserPriv.Admin_UserPriv_Agg	An administrator aggregates information that indirectly reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.	O.Limit_ObserveRoles O.Prevent_Link O.Prevent_Observe	5,3
T.Admin_UserPriv.Admin_UserPriv_Col	An administrator reads information collected by the IT system or product that reveals the identity (or other privacy related information) of user(s) in violation of user privacy policy.	O.Prevent_AskPrivInfo O.Permit_Aliases O.Permit_Anonymity	5,3,1 (in this scenario private information will probably be printed document).
T.Component_Failure.Ext_Crypto_Failure	The TOE fails to provide adequate key management or operation due to failure of external cryptographic support	O.Crypto_Extern_Depend	4
T.Component_Failure.Hardware_Flaw	System use uncovers a hardware flaw in a critical system component.	O.Fail_Secure O.Fault_Tolerance	7
T.Component_Failure.Phys_CompFail_Res	A system allocates so many resources that not enough are left for a critical component to function correctly.	O.Priority_Of_Service O.Resource_Quotas	7
T.Component_Failure.Soft	An authorized user performs an	O.Fail_Secure	7

ware_Flaw	operation or set of operations, exercising a software flaw in a security-critical component.	O.Fault_Tolerance	
T.Component_Failure.TSF _Err_Conf_Crypto	The TSF accidentally releases sensitive plaintext data, red keys, or other cryptographic assets to an inappropriate audience.	O.Crypto_Data_Sep O.Crypto_Dsgn_Impl O.Crypto_Key_Man O.Crypto_Modular_Dsgn O.Crypto_Operation O.Crypto_Self_Test O.Crypto_Test_Reqs O.Fail_Secure O.Secure_State	7,4
T.Dev_Flawed_Code.Dev_ FC_Attr_Interp	The security-critical (TSF) components inconsistently interpret audit data attributes exchanged with another trusted IT product.	O.Integrity_Attr_Exch	In this scenario the other IT product are the operating system, the printer's software (drivers) and the software that produce the document that need to be print.
T.Dev_Flawed_Code.Dev_ FC_Buff_Not_Clr	The system leaves user information in a system buffer for view by another unauthorized user.	O.No_Residual_Info	3, in this scenario user information can be last user that used a printed, name or size of the last printed document.
T.Dev_Flawed_Code.Dev_ FC_Ctrl_Data	A security-critical (TSF) component incorrectly modifies control data regarding a user process.	O.Integ_Sys_Data_Int	Because outputware application need other application to actually produce something (the operating system, the document editing software, etc…), this threat need special attention.
T.Dev_Flawed_Code.Dev_ FC_Data_Export	The system incorrectly exchanges system data with another trusted system.	O.Integ_Sys_Data_Ext	This threat is relevant because in our context other system can be the printer software. The output need to be same as the intended output.
T.Dev_Flawed_Code.Dev_ FC_Recovery	A system failure may alter the behavior of the system's security functions in such a way that, upon recovery, it no longer properly enforces its security policy (TSP).	O.Secure_State	7, today's hardware need electricity, this simple fact force us to think about those kind of problem.
T.Dev_Flawed_Code.Dev_ FC_Self_Protect	Software developer or hacker modifies system security functions	O.Correct_Operation O.Sys_Self_Protection	[Attack]

	resulting in a loss of security protection.		
T.Dev_Flawed_Code.Dev_FC_Trap_Door	The system developer creates a secret back door in the system (TOE) that allows covert access by the developer. This allows the developer to collect information, monitor user actions, modify the operation of the TOE, or just make unauthorized use of the TOE.	O.Audit_Account O.Audit_Admin_Role O.Code_Signing O.I&A_User O.Source_Code_Exam	[Attack]
T.Dev_Flawed_Code.Ext_Crypto_Failure	The TOE fails to provide adequate key management or operation due to failure of external cryptographic support	O.Crypto_Extern_Depend	4 (note that this threat is different than T.Component_Failure.Ext_Crypto_Failure that consider failure in external cryptography software, here we considered misuse of a cryptography library).
T.Failure_DS_Comp.Failure_DS_Comm	Failure of a communications function severs communications between security-critical (TSF) components.	O.Fault_Tolerance O.Integrity_Data_Rep O.Trusted_DS_Recov	2,7
T.Hack_AC.Hack_AC_Code_Vul	The hacker can use vulnerabilities found in system or application code to break into a system undetected.	O.Apply_Code_Fixes O.Audit_Deter_Misuse	2,7
T.Hack_AC.Hack_AC_Weak	The system access control mechanism(s) or user attributes are weak and can be broken or the implementation methods of the system access control causes the weakness.	O.Hack_Limit_Sessions O.Trusted_Path	3, this threat will have impact on the tradeoff between usability (how easy it's to get the document printed) and security (how to avoid anyone to print a document).
T.Hack_Avl_Resource.Hack_Comm_Overload	The unauthorized use of communication resources by a hacker causes a denial or delay in service to legitimate operations within the TOE scope of control. This would include the excess bandwidth utilization, leading to the TOE's inability to perform it's security functions.	O.Audit_Generation O.Data_Imp_Exp_Control O.Hack_Limit_Sessions O.Hack_Traffic_Control O.Priority_Of_Service O.Resource_Quotas	7,2,1 Lot of printer, specifically large volume printers use "classic" network interface. [Attack]
T.Hack_Avl_Resource.Hack_Prcsr_Overload	Hacker causes system task overload resulting in denial of service. The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The hacker invokes processing functions in	O.Audit_Generation O.Hack_Limit_Sessions O.Hack_Traffic_Control O.Priority_Of_Service O.React_Discovered_Atk O.Resource_Quotas	[Attack] 7,1 In our context, one of the critical resource managed by the software are printer.

	association with unauthorized activity that leads to overburdening processing resources on the TOE.		
T.Hack_Avl_Resource.Hack_Stg_Overload	A hacker initiates processes that tax the amount of storage available in the system (TOE). Such would be the case when a hacker floods the TOE with e-mails.	O.Audit_Generation O.Guarantee_Audit_Stg O.Hack_Limit_Sessions O.Hack_Traffic_Control O.Manage_TSF_Data O.Priority_Of_Service O.Resource_Quotas	[Attack] 7,1 In our context, storage can be abused by flooding the printing queue with lot of large document.
T.Hack_Comm_Eavesdrop.Hack_CommEaves_Eman	An outsider uses special equipment to capture emanations off the communications line.	O.Data_Exchange_Conf	[Attack] 7,2,1
T.Hack_Comm_Eavesdrop.Hack_CommEaves_Intrc	An outsider who is not an intended recipient intercepts user data communications.	O.Data_Exchange_Conf	[Attack] 7,2,1 In our context remember that data communication result are often printed documents.
T.Hack_Comm_Eavesdrop.Hack_CommEaves_Tap	An outsider uses a device to physically tap the communications line.	O.Comm_Line_Protection O.Tamper_ID O.Tamper_Resistance	7,2,1
T.Hack_Crypto.Hack_Crypto_ChsnCy	The attacker discovers an unknown encryption key.	O.Encryption_Access O.Encryption_Prohibit O.Robust_Encryption	4
T.Hack_Crypto.Hack_Crypto_ChsnPln	An attacker discovers an unknown encryption key by choosing a set of plaintexts and causing the corresponding set of ciphertexts to be generated.	O.Encryption_Access O.Encryption_Prohibit O.Robust_Encryption	4
T.Hack_Crypto.Hack_Crypto_ChsnTxt	An attacker discovers an encryption key by choosing samples of both plaintext and ciphertext, and causing them to be encrypted and decrypted, respectively, using a known algorithm and the unknown key.	O.Crypto_Data_Sep O.Encryption_Access O.Encryption_Prohibit O.Robust_Encryption	4
T.Hack_Crypto.Hack_Crypto_Cypher	An attacker discovers the plaintext that corresponds to the given ciphertext, knowing only the encryption algorithm being used. The attacker has no plain-text examples to work from, nor does he/she know the encryption key that was used.	O.Encryption_Access O.Robust_Encryption	4
T.Hack_Crypto.Hack_Crypto_PlnTxt	An attacker discovers an encryption key by comparing corresponding plaintext and ciphertext samples.	O.Encryption_Access O.Robust_Encryption	4
T.Hack_Crypto.Hack_Phys_Cnf_Eman	An attacker collects unintended system emanations, interprets them, and thus retrieves information that is being processed by the system.	O.EMSEC_Design O.IntelEman_Contain O.IntelEman_Control	4
T.Hack_Masq.Hack_Masq	A hacker captures the interactive	O.Audit_Gen_User	4,3,5 In our context

_Hijack	session of an authorized user. The hacker now appears as a legitimate user and can perform any action allowed to that user, including reading or modifying sensitive data.	O.Trusted_Path	the threat can be summarized as: let unauthorized users use printers. [Attack]
T.Hack_Masq.Hack_Masq _Uwkstn	An individual takes advantage of an unattended but active workstation to perform operations in the name of the logged-in user. Such operations may include some operations that the attacker is not normally allowed to perform.	O.Audit_Gen_User O.Screen_Lock O.Session_Termination O.Trusted_Path O.User_Guidance	4,3,5,2 In our context the threat can be summarized as: let unauthorized users use printers. [Attack]
T.Hack_Masq.Hack_Masq _Wauth	Services are provided to a user application without adequate authentication of the client requesting the service. This would permit someone to receive services for which they are not authorized. However, the server would see them as a legitimate user, which is why this is classified as a masquerade attack.	O.Audit_Generation O.User_Auth_Enhanced O.User_Auth_Multiple	4,3,5,2 In our context the threat can be summarized as: let unauthorized users use printers. [Attack]
T.Hack_Msg_Data.Hack_ MsgData_RcvTSF	Security-critical (TSF) data is modified in transit from a remote trusted site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.	O.TSF_Rcv_Err_ID_Loc O.TSF_Rcv_Err_ID_Rem O.TSF_Rcv_Err_Rcvr_Lo c O.TSF_Rcv_Err_Rcvr_Re m	2 for instance: modification of user authentification during the authentification phase that would have impact on user printing quotas.
T.Hack_Msg_Data.Hack_ MsgData_RcvUsr	A hostile outsider modifies message data in route to the system. Alternatively, errors in the communications infrastructure modify the message.	O.Rcv_MsgMod_ID O.Rcv_MsgMod_Rcvr	1,2 for instance: modify the feedback about document queue state, page printed, document owner,…
T.Hack_Msg_Data.Hack_ MsgData_SndTSF	Security-critical (TSF) data is modified in transit to a remote site, either accidentally by the communications infrastructure or deliberately by a hostile outsider.	O.TSF_Snd_Err_ID_Loc O.TSF_Snd_Err_ID_Rem O.TSF_Snd_Err_Rcvr_Lo c O.TSF_Snd_Err_Rcvr_Re m	1,2
T.Hack_Msg_Data.Hack_ MsgData_SndUsr	A hostile outsider modifies message data in route to a remote site. Alternatively, errors in the communications infrastructure modify the message.	O.Snt_MsgMod_ID O.Snt_MsgMod_Rcvr	1,2 for instance: modify the document that will be printed using the TOE.
T.Hack_Phys.Hack_Phys_ Avl_Eman	System emissions, typically electromagnetic radiation, disrupt electronic circuits in nearby equipment, causing them to fail or behave erratically.	O.InterferEman_Control	1,2 for instance: modify the document that will be printed using the TOE.
T.Hack_Phys.Hack_Phys_	An attacker collects unintended	O.EMSEC_Design	1,2,7 for instance:

Cnf_Eman	system emanations, interprets them, and thus retrieves information that is being processed by the system.	O.IntelEman_Contain O.IntelEman_Control	getting information about what document are printed by a given user. [Attack]
T.Hack_Phys.Hack_Phys_Crypto	Physical attack causes damage to cryptographic functions and/or release of cryptographic assets	O.Tamper_ID O.Tamper_Resistance	4,7 [Attack]
T.Hack_Phys.Hack_Phys_Damage	Hacker physically attacks the system, causing physical damage and loss of security protection.	O.Tamper_ID O.Tamper_Resistance	7 One of the most easy application of this threat would be to cut printer's wire. [Attack]
T.Hack_Social_Engineer.Hack_SocEng_Password	A hacker persuades a user or administrator to reveal his password, giving the hacker access to the person's account privileges.	O.Limit_Mult_Sessions O.User_Auth_Enhanced	This threat cover "social engineered" type of attack. [Attack]
T.Hack_Social_Engineer.Hack_SocEng_SysInfo	A hacker persuades a user or administrator to reveal information about system operational procedures, auditing and known flaws.	O.Admin_Guidance O.Audit_Unusual_User O.Identify_Unusual_Act O.User_Guidance	This threat cover "social engineered" type of attack. [Attack]
T.Malicious_Code.Mal_Code_Hack_Downld	A perpetrator disseminates malicious code via push or pull mechanism.	O.Clean_Obj_Recovery O.Code_Signing O.General_Integ_Checks O.Input_Inspection O.Obj_Protection O.Remote_Execution	7 [Attack]
T.Malicious_Code.Mal_Code_Hack_Exe	A perpetrator executes malicious code either remotely or locally.	O.Admin_Code_Val O.Clean_Obj_Recovery O.Code_Signing O.General_Integ_Checks O.I&A_User_Action O.Isolate_Executables O.Remote_Execution O.Trusted_Path O.Trusted_Path&Channel	7 [Attack]
T.Malicious_Code.Mal_Code_IT_Download	An IT device accidentally transfers or downloads malicious code to itself or other device that it can influence.	O.Clean_Obj_Recovery O.Code_Signing O.General_Integ_Checks O.Input_Inspection O.Obj_Protection	7. For instance: a flaw in the OS on which the TOE runs.
T.Malicious_Code.Mal_Code_IT_Exe	An IT device under normal operations enters a state required to execute the malicious code.	O.Admin_Code_Val O.Clean_Obj_Recovery O.Code_Signing O.General_Integ_Checks O.I&A_User_Action O.Isolate_Executables	7
T.Malicious_Code.Mal_Code_Usr_Downld	An authorized user accidentally downloads malicious code.	O.Clean_Obj_Recovery O.Code_Signing O.Input_Inspection O.Obj_Protection	1,7 This threat becomes more important if the TOE applies modification on the printed documents (that can the case in

			our 3 usage model). Remember that document are sent to the outputware in order to get printed.
T.Malicious_Code.Mal_Code_Usr_Exe	An authorized user executes malicious code accidentally.	O.Admin_Code_Val O.Clean_Obj_Recovery O.Code_Signing O.General_Integ_Checks O.I&A_User_Action O.Isolate_Executables	1,7 This threat is the logic sequel to T.Malicious_Code. Mal_Code_Usr_Do wnld (see above). In our context executing the code can be done by actually printing the document. Because the current printer use postscript programming language to describe the document, it's possible to perform some malicious action using this programming language.
T.Power_Disrupt.Power_Disrupt_Reset	An unintentional, malicious, or environmentally caused power reset occurs, resulting in the loss of critical information or the system to enter a non-secure state.	O.Atomic_Functions O.Trusted_Recovery	7,2 For instance: blackout during printing.
T.Repudiate_Receive.Repudiate_Rcvr_Int	A local, authorized user receives a message from another local user via the system, and then denies having received it. This typically affects the sender of the message who is counting on responsibilities associated with receipt of the message.	O.NonRepud_Locals_Rcvd	2 for instance: an administrator informs the user that the next printed copies will have an additional cost.
T.Repudiate_Receive.Repudiate_Rcvr_Local	A local, authorized user receives a message from another user at a remote trusted product, and then denies having received it.	O.NonRepud_Assess_Recd O.NonRepud_Gen_Recd	1,2 In our context, most of the feedback information are coming from the printer (print state, paper format allowed, …), the printer represent the remote user.
T.Repudiate_Receive.Repudiate_Rcvr_Rem	A local, authorized user sends a message to another user at a remote trusted product who then denies having received it.	O.NonRepud_Assess_Recd O.NonRepud_Gen_Recd	1,2 for instance: an administrator send information about printing cost.
T.Repudiate_Send.Repudiate_Send	A local, authorized user sends a message to another local user via the system, and then denies	O.NonRepud_Locals_Sent	1,2 for instance: an user orders 200 copies of the same

	having done it. This affects the recipient of the message as well as any resources allocated or modified by the recipient in response to the message.		document then deny it.
T.Repudiate_Transact.Repudiate_Trans	An authorized user participates in a transaction by responding to system/application prompts and then denies that the dialogue took place. The user and system/application are collocated.	O.I&A_Transaction O.NonRepud_Locals_Rcvd O.NonRepud_Locals_Sent	1,2 for instance: an user prints 200 copies of the same document then deny it.
T.Spoofing.Hack_Spoof_Login	An attacker simulates the system's login program and runs it at an open terminal or workstation in order to capture a legitimate user's authentication data.	O.Trusted_Path O.User_Auth_Enhanced	2 this threat represent the classic "replay" attach. [Attack]
T.Spoofing.Hack_Spoof_MsgHdr	An attacker may modify protocol headers such that a user believes the communication is coming from a source that is different from where it was actually sent.	O.Comm_Trusted_Channel O.Crypto_Comm_Channel O.Integrity_Attr_Exch O.NonRepudiate_Sent	2
T.User_Abuse_Conf.User_Abuse_Conf_Disk	A user collects sensitive or proprietary information and improperly removes it from the system by putting it on removable media.	O.Admin_Guidance O.Audit_Account O.Data_Export_Control	3 for instance: importing data or document from a printing queue and exporting them to removable media
T.User_Abuse_Conf.User_Abuse_Conf_Steg	An authorized user hides sensitive information in an innocuous-appearing file, for the purpose of covertly passing it to an unauthorized party. The hidden data is undetectable to anyone using the file for its intended purpose, but can be recovered using special techniques.	O.Admin_Code_Val O.Admin_Code_Val_Sten O.Export_Control O.Standard_Output_Pres	1 for instance: an user hide sensitive data inside a larger printed document.
T.User_Collect.User_Collect_Browse	An authorized user abuses granted authorizations by browsing files in order to collect data.	O.Audit_Account O.Info_Flow_Control O.User_Defined_AC	7 for instance: browsing the current printing queue to collect data about someone.
T.User_Collect.User_Collect_Deceive	An authorized user steals authentication data by emulating a login procedure on an active terminal.	O.Access_History O.Trusted_Path	2
T.User_Collect.User_Collect_Deduce	An authorized user abuses granted authorizations by repeatedly accessing aggregate data in order to deduce specific, sensitive data.	O.Audit_Generation O.Info_Flow_Control O.User_Defined_AC	1 Useful information could be retrieved from printing queue (Who print the most? When? Etc.)
T.User_Collect.User_Collect_Eaves	An authorized user abuses granted authorizations by eavesdropping on communication lines in order	O.Data_Exchange_Conf O.Integ_User_Data_Int O.Security_Roles	1,2

	to collect data.		
T.User_Collect.User_Collect_Residue	An authorized user collects residual data from public objects after prior usage.	O.No_Residual_Info	1 in our context, attention is needed on the fact that residual info of the process can be printed document. It can be the whole document or only the header page of the printer.
T.User_Err_Conf.Hack_Ext_CryptoAsset	Cryptographic assets are mishandled after they leave the TOE, either in transit or while residing on stored media.	O.Crypto_Import_Export O.Crypto_Manage_Roles	4
T.User_Err_Conf.User_Err_Conf_Class	An authorized user presents confidential or classified information to a recipient, indicating that it is less sensitive than it really is, thereby encouraging the recipient to pass it along to other potentially inappropriate recipients.	O.AC_Label_Export	1,3 for instance: printing bills on a public printer because the document wasn't marked as confidential.
T.User_Err_Conf.User_Err_Conf_Crypto	User error causes release of cryptographic assets to unauthorized recipients.	O.Crypto_AC O.Crypto_Key_Man O.I&A_Domain O.I&A_User_Action	3, This threat is a generic threat for every system using authentification.
T.User_Err_Conf.User_Err_Conf_Exp	An authorized user exposes or exports data in violation of export control policy. The data may be private or classified, the recipient is not authorized to receive it.	O.User_Conf_Prevention	1,3 in this context, "expose" can be done by printing a document.
T.User_Err_Inaccess.User_Err_Delete	An authorized user accidentally deletes user data.	O.Rollback O.User_Guidance	3 for instance: a user deletes the wrong document in the printing queue.
T.User_Err_Inaccess.User_Err_Mod_Attr	An authorized user erroneously modifies the initial security attributes of user data, which makes the data inaccessible.	O.Security_Attr_Mgt O.User_Guidance	3 for instance: a user modifies document properties in such a way that it can't be printed anymore.
T.User_Err_Inaccess.User_Err_Set_Attr	An authorized user erroneously sets the initial security attributes of user data, which makes the data inaccessible.	O.Security_Attr_Mgt O.User_Guidance	3, for instance a user set a document to be printed using wrong paper format causing the document to stay blocked in the printer queue.
T.User_Err_Integrity.Hack_Ext_CryptoAsset	Cryptographic assets are mishandled after the leave the TOE, either in transit or while residing on stored media.	O.Crypto_Import_Export O.Crypto_Manage_Roles	4
T.User_Err_Integrity.User_Err_AttrXpt	An authorized user presents incorrect information, indicating	O.AC_Label_Export	4,3 for instance: printing a

	to the recipient that it is correct, thereby encouraging the recipient to make unwarranted use of the information.		document in an incompatible format.
T.User_Err_Integrity.User_Modify_Data	An authorized user modifies or deletes user data in violation of organizational policy.	O.Audit_Generation O.Info_Flow_Control O.User_Defined_AC	4,3 for instance modifying print queue state.
T.User_Err_Slf_Protect.User_Err_MsngAttrXpt	An authorized user deliberately or accidentally exports data so that the data is not accompanied by required handling information.	O.AC_Label_Export	1,3 for instance sending document to the outputware without using the adequate software.
T.User_Err_Slf_Protect.User_Err_Object_Attr	An authorized user sets an object's security attributes inappropriately, misdirecting its use. The misdirection may allow unauthorized reading or modification, or it may prohibit authorized reading or modification.	O.AC_Label_Export O.Obj_Attr_Integrity	1,3 for instance a user print the document with wrong privacy option, therefore it get printed on the wrong printer.
T.User_Misuse_Avl_Resc.User_Comm_Overload	An authorized user exceeds the authorized use of communication resources during the system (TOE) operation. This causes a denial or delay in service to legitimate operations within the TOE scope of control.	O.Audit_Generation O.Data_Imp_Exp_Control O.Limit_Comm_Sessions O.Priority_Of_Service O.Resource_Quotas	1,3 for instance: a user sending the same large document many times.
T.User_Misuse_Avl_Resc.User_ErrAvl_AudExhst	An authorized user's actions generate so many audit records that audit storage space is exhausted and the system subsequently denies further service until audit storage becomes available.	O.Audit_Loss_Respond O.Guarantee_Audit_Stg O.Manage_TSF_Data	1,3,6
T.User_Misuse_Avl_Resc.User_Obst_Res_Use	An authorized user obstructs the use resources by unauthorized modification of data file, communication channel, or object security attributes.	O.Manage_Res_Sec_Attr O.Priority_Of_Service O.Tamper_ID O.Tamper_Resistance	1,3 for instance: a user flooding the network with unuseful data in a way that prevent TOE to communicate with the other part of the network.
T.User_Misuse_Avl_Resc.User_Prcsr_Overload	The system (TOE) has been over-tasked and can not complete the assigned tasking at all or in an expected amount of time. The user invokes processing functions in association with unauthorized activity that leads to overburdening processing resources on the TOE.	O.Audit_Generation O.Limit_Comm_Sessions O.Priority_Of_Service O.Resource_Quotas	1,3 this problem can occur if too many users are trying to print documents at the same time.
T.User_Misuse_Avl_Resc.User_Stg_Overload	An authorized user's unauthorized use of data storage causes a shortage of disk space for other users.	O.Audit_Generation O.Limit_Comm_Sessions O.Priority_Of_Service O.Resource_Quotas	1,2,3 for instance: a user that use space on the TOE's host machine for other purpose.

T.User_Modify.User_Modify_Audit	An authorized user modifies audit data or audit attributes to avoid accountability.	O.Audit_Gen_User O.Audit_Generation O.Audit_Protect O.Security_Roles	1,3, for instance: a user try to modify his printing quota in order to be able to print more page.
T.User_Modify.User_Modify_Auth	An authorized user changes the authentication data of another user without first masquerading as that user, in a manner that is not consistent with organizational security policy.	O.Audit_Account O.Security_Data_Mgt	1,3
T.User_Modify.User_Modify_Data	An authorized user modifies or deletes user data in violation of organizational policy.	O.Audit_Generation O.Info_Flow_Control O.User_Defined_AC	1,3
T.User_Modify.User_Modify_TSFData	User modifies or deletes TSF data undermining security protection.	O.Audit_Generation O.Config_Management O.General_Integ_Checks O.Info_Flow_Control O.Integ_Sys_Data_Int O.Integrity_Practice O.Maintain_Sec_Domain O.Reference_Monitor O.User_Defined_AC	1,3 this threat can be seen if the user is hostile, therefore it's considered as an attack [Attack]
T.User_Send.User_Abuse_Conf_Steg	An authorized user hides sensitive information in an innocuous-appearing file, for the purpose of covertly passing it to an unauthorized party. The hidden data is undetectable to anyone using the file for its intended purpose, but can be recovered using special techniques.	O.Admin_Code_Val O.Admin_Code_Val_Sten O.Export_Control O.Standard_Output_Pres	1,3 this threat can be seen if the user is hostile, therefore it's considered as an attack [Attack]
T.User_Send.User_Send_Conf	An authorized user abuses granted authorizations and violates export control policy by sending data to a recipient who is not authorized to receive the data.	O.Audit_Generation O.Integ_Data_Mark_Exp	1,3 this threat can be seen if the user is hostile, therefore it's considered as an attack [Attack]
T.User_Send.User_Send_Integrity	An authorized user deliberately exports data inappropriately, with the result that there is a lack of required quality control on the exported data.	O.Audit_Generation O.Integ_Data_Mark_Exp	1,3 this threat can be seen if the user is hostile, therefore it's considered as an attack [Attack]

1.3 Policies

Name	Description	Discussion
P.Changes to security data by authorized personnel	Provide mechanisms to assure that changes to security related data are executed only by authorized personnel.	Depending of the presence of an operator, this policy will impact 2 or 3 kinds of users: classic users, administrators and operator.
P.Individual accountability	The system shall provide individual accountability for auditable actions.	In our context auditable actions are all kind of document output but also security related

		information (user login, ...).
P.Audit data generation with identity	The system shall provide the capability to ensure that all audit records include enough information to determine the date and time of action, the system locale of the action, the system entity that initiated or completed the action, the resources involved, and the action involved.	
P.Protected audit data storage	The system shall protect the contents of the audit trails against unauthorized access, modification, or deletion.	
P.Notification of threats and vulnerabilities	Notification of threats and vulnerabilities shall be addressed.	
P.Notification of data content changes	Notify user of the time and date of the last modification of data.	This is a generic policy that needs to be instantiated to say which data needs notification.
P.Implement operational configuration management	A configuration management plan shall be implemented by the system. The system shall implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware). The system shall implement strong integrity mechanisms (integrity locks, encryption).	
P.Documented recovery	The system shall provide procedures and features to assure that system recovery is done in a trusted and secure manner. Any circumstances that could result in an untrusted recovery shall be documented.	
P.Labeling data	The system shall provide security parameters associated with information exchanged between systems.	
P.User identification and authentication	The system shall provide Identification and authentication (I&A) procedures which uniquely identify and authenticate users.	
P.Strong integrity mechanisms	The system shall implement strong integrity mechanisms (integrity locks, encryption).	
P.Operational integrity system function testing	Provide system functional tests to periodically test the integrity of the hardware and code running system functions.	
P.Security throughout lifecycle	Security shall be addressed throughout the system's lifecycle.	
P.Preventive maintenance	The system administrators shall provide preventive maintenance.	
P.Malicious code prevention	Procedures and mechanisms to prevent the introduction of malicious code into the system shall be provided.	
P.Privileged user access	The system shall function so that each user has access to all of the information and functions that the user requires to perform duties, but no more.	
P.Non-repudiation	The system shall provide non-repudiation	An example of a non-

capabilities	capabilities.	repudiation mechanism is an implementation of digital signatures.
P.Privileged user documentation	Documentation shall include guides or manuals for the system's privileged users.	
P.User screen locking	The system shall provide a screen lock mechanism.	
P.Assurance of effective storage integrity	The system shall provide assurance that storage integrity is effective.	
P.System access banners	The system shall notify users prior to gaining access that the user's actions may be monitored and recorded, that using the system consents to such monitoring, and that unauthorized use may result in criminal or civil penalties.	
P.Validation of security function integrity	Features and procedures to validate the integrity and the expected operation of the security-relevant software, hardware, and firmware shall be provided by the system.	
P.System backup procedures	Provide the capability to restore the system to a secure state after discontinuities of system operations.	
P.Restoration with minimal loss	The system shall provide backup procedures to allow restoration of the system with minimal loss of service or data.	
P.Effective backup restoration	The system shall provide procedures to ensure both the existence of sufficient backup storage capability and effective restoration (incremental and complete) of the backup data.	
P.Backup protection and restoration	The system shall provide appropriate physical and technical protection of the backup and restoration hardware, firmware, and software.	
P.Protection from security function modification	Provide features or procedures for protection of the system from improper changes.	
P.Trusted system recovery	Provide procedures and features to assure that system recovery is done in a trusted and secure manner.	
P.Physical tampering detection and notification	The system shall detect physical tampering and notify the appropriate authority.	
P.Enhanced user identification and authentication	The system shall require the use of enhanced authentication for privileged users who either reside outside of the system's perimeter or whose communications traverse data lines outside of the system's perimeter.	
P.Encryption of transmitted user data	The system shall provide data transmission using an encryption mechanism appropriate for the sensitivity of the data.	
P.Protection of stored user data	The system shall provide appropriate storage, continuous personnel access control storage, or encrypted storage of data based on the sensitivity of the data.	
P.Protection of	The system shall provide a protected distribution	

transmitted user data	system for data transmitted.	
P.Discretionary access control	The system shall provide a Discretionary Access Control (DAC) function (i.e., a user can grant access authorization to other users for data they control).	
P.General user documentation	Documentation shall include a user's guide for the general user.	

1.4 Objectives

Name	Description	Covers
O.AC_Admin_Limit	Design administrative functions in such a way that administrators do not automatically have access to user objects, except for necessary exceptions. For an accounts administrator, the necessary exceptions include allocation and de-allocation of storage. For an audit administrator, the necessary exceptions include observation of audited actions. In general, the exceptions tend to be role specific.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC T.Admin_Hostile_Modify.Adm_Hstl_Mod_DataAps T.Admin_Hostile_Modify.Adm_Hstl_Mod_SEP
O.AC_Label_Export	Provide object security attributes in exported data with moderate to high effectiveness. The attributes are those associated with specific security function policies.	T.User_Err_Slf_Protect.User_Err_MsngAttrXpt T.User_Err_Integrity.User_Err_AttrXpt T.User_Err_Conf.User_Err_Conf_Class T.User_Err_Slf_Protect.User_Err_Object_Attr
O.Access_History	Display information related to the most recent successful and unsuccessful attempts to establish a user session, once a user successfully establishes a user session.	T.User_Collect.User_Collect_Deceive
O.Adm_Limits_Bindings	Limit the administrator from modification of user-subject bindings in an effort to deter users acting without accountability.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_USB
O.Adm_User_Att_Mod	Deter the administrator from maliciously modifying users' attributes. Such modifications could allow unauthorized user actions or denial of service to a legitimate user.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_UsrAttr
O.Admin_Code_Val	Validate executable objects prior to allowing execution. Validation needs to be done by someone with an expertise to recognize malicious code and the authority and means to prevent its execution.	T.User_Send.User_Abuse_Conf_Steg T.Malicious_Code.Mal_Code_Usr_Exe T.User_Abuse_Conf.User_Abuse_Conf_Steg T.Malicious_Code.Mal_Code_IT_Exe T.Malicious_Code.Mal_Code_Hack_Exe
O.Admin_Code_Val_Sten	Validate exported objects for absence of steganographic content prior to allowing exportation. Validation needs to be done by someone with an expertise to recognize hidden content and the authority and means to prevent its export.	T.User_Abuse_Conf.User_Abuse_Conf_Steg T.User_Send.User_Abuse_Conf_Steg

O.Admin_Guidance	Deter administrator errors by providing adequate administrator guidance.	T.Admin_Err_Commit.Admin_Err_Authentic T.User_Abuse_Conf.User_Abuse_Conf_Disk T.Admin_Err_Commit.Admin_Err_Resource T.Admin_Err_Commit.Admin_Err_User_Attr T.Admin_Err_Commit.Admin_Err_AC_Policy T.Admin_Err_Commit.Admin_Err_Audit T.Admin_Err_Omit.Admin_Err_Update T.Hack_Social_Engineer.Hack_SocEng_SysInfo T.Admin_Err_Commit.Admin_Err_Info T.Admin_Err_Commit.Admin_Err_Sys_Entry
O.Apply_Code_Fixes	Apply patches to fix the code when vulnerabilities in code allow unauthorized and undiscovered access.	T.Hack_AC.Hack_AC_Code_Vul
O.Atomic_Functions	Recover automatically to a consistent, secure state if a security function does not complete successfully in the presence of certain types of failures.	T.Power_Disrupt.Power_Disrupt_Reset
O.Aud_Sys_Entry_Parms	Deter an administrator from changing system entry parameters to allow an unauthorized user access to organizational assets to which they are forbidden.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_SEP
O.Audit_Account	Provide information about past user behavior to an authorized user through system mechanisms. Specifically, during any specified time interval, the system is able to report to a user acting in an identified audit role selected auditable actions that a user has performed, and as a result, what auditable objects were affected and what auditable information was received by that user.	T.User_Modify.User_Modify_Auth T.Admin_Err_Omit.Admin_Err_Update T.User_Collect.User_Collect_Browse T.User_Abuse_Conf.User_Abuse_Conf_Disk T.Admin_Err_Commit.Adm_Err_Crypto T.Admin_Err_Omit.Adm_Err_Crypto T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC T.Dev_Flawed_Code.Dev_FC_Trap_Door
O.Audit_Admin_Role	Deter modification or destruction of audit data through the creation of an audit-administration role.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_DataAps T.Admin_Hostile_Modify.Adm_Hstl_Mod_SEP T.Admin_Err_Commit.Admin_Err_Audit T.Dev_Flawed_Code.Dev_FC_Trap_Door T.Admin_Hostile_Modify.Adm_Hstl_Audit_Dstr T.Admin_Hostile_Modify.Adm_

		Hstl_Mod_IFC
O.Audit_Deter_Misuse	Audit system access to discover system misuse and provide a potential deterrent by warning the user.	T.Hack_AC.Hack_AC_Code_Vul
O.Audit_Gen_User	Provide individual accountability for audited events. Uniquely identify each user so that auditable actions can be traced to a user.	T.Hack_Masq.Hack_Masq_Hijack T.Hack_Masq.Hack_Masq_Uwkstn T.User_Modify.User_Modify_Audit
O.Audit_Generation	Record in audit records: date and time of action, location of the action, and the entity responsible for the action.	T.User_Misuse_Avl_Resc.User_Comm_Overload T.User_Modify.User_Modify_Audit T.User_Modify.User_Modify_Data T.User_Modify.User_Modify_TSFData T.User_Misuse_Avl_Resc.User_Stg_Overload T.User_Misuse_Avl_Resc.User_Prcsr_Overload T.User_Err_Integrity.User_Modify_Data T.User_Send.User_Send_Conf T.User_Send.User_Send_Integrity T.User_Collect.User_Collect_Deduce T.Hack_Masq.Hack_Masq_Wauth T.Hack_Avl_Resource.Hack_Prcsr_Overload T.Hack_Avl_Resource.Hack_Stg_Overload T.Hack_Avl_Resource.Hack_Comm_Overload
O.Audit_Loss_Respond	Respond to possible loss of audit records when audit trail storage is full or nearly full.	T.Admin_Err_Commit.Admin_Err_Audit T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC T.User_Misuse_Avl_Resc.User_ErrAvl_AudExhst
O.Audit_Protect	Protect audit records against unauthorized access, modification, or deletion to ensure accountability of user actions.	T.Admin_Hostile_Modify.Adm_Hstl_Audit_Dstr T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC T.User_Modify.User_Modify_Audit T.Admin_Hostile_Modify.Adm_Hstl_Mod_IFC T.Admin_Err_Commit.Admin_Err_Audit
O.Audit_Unusual_User	Audit unusual user activity.	T.Hack_Social_Engineer.Hack_SocEng_SysInfo
O.Change_Control_Users	Notify users of changes to data content in order to make any adjustments to their own data.	

O.Clean_Obj_Recovery	Recover to a viable state after malicious code is introduced and damage occurs, removing the malicious code as part of the process.	T.Malicious_Code.Mal_Code_IT_Download T.Malicious_Code.Mal_Code_Hack_Downld T.Malicious_Code.Mal_Code_Usr_Exe T.Malicious_Code.Mal_Code_IT_Exe T.Malicious_Code.Mal_Code_Usr_Downld T.Malicious_Code.Mal_Code_Hack_Exe
O.Code_Signing	Check verification of signed downloaded code prior to execution. A well-known example is checking digital signatures on signed Java applets.	T.Malicious_Code.Mal_Code_IT_Download T.Malicious_Code.Mal_Code_Usr_Downld T.Malicious_Code.Mal_Code_Hack_Exe T.Malicious_Code.Mal_Code_Hack_Downld T.Dev_Flawed_Code.Dev_FC_Trap_Door T.Malicious_Code.Mal_Code_Usr_Exe T.Malicious_Code.Mal_Code_IT_Exe
O.Comm_Line_Protection	Protect communications lines from physical tampering.	T.Hack_Comm_Eavesdrop.Hack_CommEaves_Tap
O.Comm_Trusted_Channel	Provide a communications channel between the system and a remote trusted system for the performance of security-critical operations.	T.Spoofing.Hack_Spoof_MsgHdr
O.Config_Management	Implement a configuration management plan. Implement configuration management to assure storage integrity, identification of system connectivity (software, hardware, and firmware), and identification of system components (software, hardware, and firmware).	T.User_Modify.User_Modify_TSFData
O.Correct_Operation	Provide the ability for the authorized user to verify that the system operates as designed.	T.Dev_Flawed_Code.Dev_FC_Self_Protect
O.Crypto_AC	Restrict user access to cryptographic IT assets in accordance with a specified user access control policy.	T.User_Err_Conf.User_Err_Conf_Crypto
O.Crypto_Comm_Channel	Provide secure session establishment between the system and remote systems using encryption functions.	T.Spoofing.Hack_Spoof_MsgHdr
O.Crypto_Data_Sep	Provide complete separation between plaintext and encrypted data and between data and keys. This requires separate channels and separate storage areas. The only place any data can pass between the plaintext and encrypted data modules is in the cryptographic engine. There should be no way for plaintext keys to reach either data module and no way for data to enter the key handling module. Eencrypted keys can be handled as encrypted data, but with limited user access.	T.Hack_Crypto.Hack_Crypto_ChsnTxt T.Component_Failure.TSF_Err_Conf_Crypto

O.Crypto_Dsgn_Impl	Minimize or even eliminate design and implementation errors in the cryptographic modules and functions.	T.Component_Failure.TSF_Err_Conf_Crypto
O.Crypto_Extern_Depend	Specify security functional requirements (SFRs) that are expected to be satisfied by other software, firmware or hardware that is external to the TOE.	T.Dev_Flawed_Code.Ext_Crypto_Failure
O.Crypto_Import_Export	Protect cryptographic data assets when they are being transmitted to and from the TOE, either through intervening untrusted components or directly to/from human users.	T.User_Err_Integrity.Hack_Ext_CryptoAsset T.User_Err_Conf.Hack_Ext_CryptoAsset
O.Crypto_Key_Man	Fully define cryptographic components, functions, and interfaces. Ensure appropriate protection for cryptographic keys throughout their lifecycle, covering generation, distribution, storage, use, and destruction.	T.Admin_Err_Omit.Adm_Err_Crypto T.User_Err_Conf.User_Err_Conf_Crypto T.Component_Failure.TSF_Err_Conf_Crypto T.Admin_Err_Commit.Adm_Err_Crypto
O.Crypto_Manage_Roles	Provide one or more roles to manage cryptographic assets and attributes.	T.Admin_Err_Omit.Adm_Err_Crypto T.Admin_Err_Commit.Adm_Err_Crypto T.User_Err_Conf.Hack_Ext_CryptoAsset T.User_Err_Integrity.Hack_Ext_CryptoAsset
O.Crypto_Modular_Dsgn	Prevent errors in one part of the TOE from influencing other parts, especially cryptographic parts. To this end, noncryptographic I/O paths must be well defined and logically independent of circuitry and processes performing key generation, manual key entry, key zeroising, and similar key-related operations.	T.Component_Failure.TSF_Err_Conf_Crypto
O.Crypto_Operation	Cryptographic components, functions, and interfaces shall be fully defined.	T.Component_Failure.TSF_Err_Conf_Crypto
O.Crypto_Self_Test	Provide the ability to verify that the cryptographic functions operate as designed.	T.Component_Failure.TSF_Err_Conf_Crypto
O.Crypto_Test_Reqs	Test cryptographic operation and key management.	T.Component_Failure.TSF_Err_Conf_Crypto
O.Data_Exchange_Conf	Protect user data confidentiality when exchanging data with a remote system.	T.User_Collect.User_Collect_Eaves T.Hack_Comm_Eavesdrop.Hack_CommEaves_Intrc T.Hack_Comm_Eavesdrop.Hack_CommEaves_Eman
O.Data_Export_Control	Impose information control policies that do not allow export of specified data and/or export to specified locations.	T.User_Abuse_Conf.User_Abuse_Conf_Disk
O.Data_Imp_Exp_Control	Protect data from being sent to erroneous places and more places external to the system than allowed by the organization's security policy. Conversely the import of data into the system should be protected from illicit information or information not allowed by the organization's security policy.	T.User_Misuse_Avl_Resc.User_Comm_Overload T.Hack_Avl_Resource.Hack_Comm_Overload

O.EMSEC_Design	Design and build the system in such a way as to control the production of intelligible emanations within specified limits.	T.Hack_Phys.Hack_Phys_Cnf_Eman T.Hack_Crypto.Hack_Phys_Cnf_Eman
O.Encryption_Access	Deter cryptoanalysis of ciphertext by denying unauthorized access to encrypted objects.	T.Hack_Crypto.Hack_Crypto_ChsnCy T.Hack_Crypto.Hack_Crypto_Cypher T.Hack_Crypto.Hack_Crypto_ChsnTxt T.Hack_Crypto.Hack_Crypto_ChsnPln T.Hack_Crypto.Hack_Crypto_PlnTxt
O.Encryption_Prohibit	Prohibit the transmission of a ciphertext message over any network where the corresponding plaintext might be available. Include cryptographic padding (i.e., random plaintext) to hide the correspondence between segments of real ciphertext and their known plaintexts.	T.Hack_Crypto.Hack_Crypto_ChsnTxt T.Hack_Crypto.Hack_Crypto_ChsnCy T.Hack_Crypto.Hack_Crypto_ChsnPln
O.Export_Control	Sanitize data objects that may contain hidden data when they are exported from the TOE in order to inhibit steganographic smuggling.	T.User_Abuse_Conf.User_Abuse_Conf_Steg T.User_Send.User_Abuse_Conf_Steg
O.External_Labels	Label or mark information for external systems to prevent the exchange of inappropriate data between systems.	
O.Fail_Secure	Preserve the secure state of the system in the event of a secure component failure.	T.Component_Failure.TSF_Err_Conf_Crypto T.Component_Failure.Hardware_Flaw T.Component_Failure.Software_Flaw
O.Fault_Tolerance	Provide fault tolerant operations for critical components and continue to operate in the presence of specific failures in one or more system components.	T.Failure_DS_Comp.Failure_DS_Comm T.Component_Failure.Hardware_Flaw T.Component_Failure.Software_Flaw
O.General_Integ_Checks	Provide periodic integrity checks on both system and user data.	T.Malicious_Code.Mal_Code_Usr_Exe T.Malicious_Code.Mal_Code_IT_Exe T.User_Modify.User_Modify_TSFData T.Malicious_Code.Mal_Code_Hack_Downld T.Malicious_Code.Mal_Code_IT_Download T.Malicious_Code.Mal_Code_Hack_Exe
O.Guarantee_Audit_Stg	Maintain audit data and guarantee space for that data.	T.Hack_Avl_Resource.Hack_Stg_Overload T.User_Misuse_Avl_Resc.User_ErrAvl_AudExhst
O.Hack_Limit_Sessions	Limit the number of sessions available to outside	T.Hack_Avl_Resource.Hack_St

	users. A hacker can initiate multiple communication sessions that could cause an overload on resources, for example, half open session starts as is seen in "SYN flood" attacks.	g_Overload T.Hack_Avl_Resource.Hack_Prcsr_Overload T.Hack_Avl_Resource.Hack_Comm_Overload T.Hack_AC.Hack_AC_Weak
O.Hack_Traffic_Control	Control (e.g. reroute or discard) hacker communication traffic to prevent potential damage.	T.Hack_Avl_Resource.Hack_Prcsr_Overload T.Hack_Avl_Resource.Hack_Stg_Overload T.Hack_Avl_Resource.Hack_Comm_Overload
O.I&A_Domain	Provide the basic I&A functions that will support user accountability.	T.User_Err_Conf.User_Err_Conf_Crypto
O.I&A_Transaction	Associate each transaction between a user and a system/application with a unique transaction ID, allowing events associated with a given transaction to be distinguished from other events involving the user and/or system/application.	T.Repudiate_Transact.Repudiate_Trans
O.I&A_User	Uniquely identify and authenticate each user of the system.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_IFC T.Admin_Err_Commit.Admin_Err_Audit T.Dev_Flawed_Code.Dev_FC_Trap_Door T.Admin_Hostile_Modify.Adm_Hstl_Mod_SEP T.Admin_Hostile_Modify.Adm_Hstl_Mod_DataAps T.Admin_Hostile_Modify.Adm_Hstl_Audit_Dstr
O.I&A_User_Action	Associate each user-requested action with the user who requested the action.	T.User_Err_Conf.User_Err_Conf_Crypto T.Admin_Err_Omit.Adm_Err_Crypto T.Malicious_Code.Mal_Code_Hack_Exe T.Malicious_Code.Mal_Code_Usr_Exe T.Malicious_Code.Mal_Code_IT_Exe T.Admin_Err_Commit.Adm_Err_Crypto
O.Identify_Unusual_Act	Identify unusual user activity on the system.	T.Hack_Social_Engineer.Hack_SocEng_SysInfo
O.Info_Flow_Control	Enforce an information flow policy whereby users are constrained from allowing access to information they control, regardless of their intent (e.g., mandatory access control). This lattice property of security attributes is commonly associated with the U.S. DoD implementations of Mandatory Access Control (MAC).	T.User_Collect.User_Collect_Deduce T.User_Modify.User_Modify_TSFData T.User_Modify.User_Modify_Data T.User_Err_Integrity.User_Modify_Data T.User_Collect.User_Collect_Browse
O.Info_Flow_Ctrl_Admin	Manage information flow control policy and functions to allow only specified administrators	T.Admin_Hostile_Modify.Adm_Hstl_Mod_IFC

	to have the ability to manipulate the information flow control.	
O.Input_Inspection	Require inspection of downloads/transfers.	T.Malicious_Code.Mal_Code_Hack_Downld T.Malicious_Code.Mal_Code_IT_Download T.Malicious_Code.Mal_Code_Usr_Downld
O.Integ_Data_Mark_Exp	Ensure that data markings are included with data that is exported to another trusted product.	T.User_Send.User_Send_Conf T.User_Send.User_Send_Integrity
O.Integ_Sys_Data_Ext	Ensure the integrity of system data exchanged externally with another trusted product by using a protocol for data transfer that will permit error detection and correction. This includes detecting and possibly correcting errors in data received and encoding outgoing data to make it possible for the receiver to detect and possibly correct errors. The method for detecting and correcting errors is based on some method (protocol) that is agreed upon by participating parties.	T.Dev_Flawed_Code.Dev_FC_Data_Export
O.Integ_Sys_Data_Int	Ensure the integrity of system data transferred internally.	T.User_Modify.User_Modify_TSFData T.Dev_Flawed_Code.Dev_FC_Ctrl_Data
O.Integ_User_Data_Int	Ensure the integrity of user data transferred internally within the system.	T.User_Collect.User_Collect_Eaves
O.Integrity_Attr_Exch	Ensure that the system correctly exchanges security-attribute information with another trusted IT product.	T.Dev_Flawed_Code.Dev_FC_Attr_Interp T.Spoofing.Hack_Spoof_MsgHdr
O.Integrity_Data/SW	Provide integrity protection for user data and software.	
O.Integrity_Data_Rep	Ensure that when system data replication occurs across the system the data is consistent for each replication.	T.Dev_Flawed_Code.Dev_FC_Replication T.Failure_DS_Comp.Failure_DS_Comm
O.Integrity_Practice	Provide system functional tests to periodically test the integrity of the hardware and code running system functions.	T.User_Modify.User_Modify_TSFData
O.IntelEman_Contain	Confine system-produced intelligible emanations to within a specified limit.	T.Hack_Crypto.Hack_Phys_Cnf_Eman T.Hack_Phys.Hack_Phys_Cnf_Eman
O.IntelEman_Control	Limit system-produced intelligible emanations to within a specified limit.	T.Hack_Crypto.Hack_Phys_Cnf_Eman T.Hack_Phys.Hack_Phys_Cnf_Eman
O.InterferEman_Control	Limit system-produced electromagnetic emanations to within a specified limit.	T.Hack_Phys.Hack_Phys_Avl_Eman
O.Isolate_Executables	Run executable code in a protected domain where the code's potential errors or malicious code will not significantly impact other system functions of other valid users of the system.	T.Malicious_Code.Mal_Code_Usr_Exe T.Malicious_Code.Mal_Code_Hack_Exe T.Malicious_Code.Mal_Code_I

		T_Exe
O.Lifecycle_Security	Provide tools, techniques, and security employed during the development phase. Detect and resolve flaws during the operational phase. Provide safe destruction techniques.	
O.Limit_Actions_Auth	Restrict the actions a user may perform before the TOE verifies the identity of the user.	T.Admin_Err_Commit.Admin_Err_Authentic
O.Limit_Comm_Sessions	Provide mechanisms to limit the number of sessions that the user can initiate, if the user initiates multiple sessions that exceed the processors ability to perform in a reliable and efficient manner. These sessions could ether be communication (TCP/IP) sessions or user login sessions.	T.User_Misuse_Avl_Resc.User_Prcsr_Overload T.User_Misuse_Avl_Resc.User_Stg_Overload T.User_Misuse_Avl_Resc.User_Comm_Overload
O.Limit_Mult_Sessions	Provide the capability to limit the number of sessions that a user may have open at one time.	T.Hack_Social_Engineer.Hack_SocEng_Password
O.Limit_ObserveRoles	Provide authorized users with the capability to observe the usage of specified services or resources as necessary to perform their duties.	T.Admin_UserPriv.Admin_UserPriv_Agg
O.Maintain_Sec_Domain	Maintain at least one security domain for system (TOE) execution to protect the TOE from interference and tampering.	T.User_Modify.User_Modify_TSFData
O.Maintenance_Access	Control access to the system by maintenance personnel who troubleshoot the system and perform system updates.	T.Admin_Err_Omit.Admin_Err_Omit_Trap
O.Maintenance_Recover	Terminate maintenance user system access privilege automatically after expiration of assigned timed interval.	T.Admin_Err_Omit.Admin_Err_Omit_Trap
O.Malicious_Code	Incorporate malicious code prevention procedures and mechanisms.	
O.Manage_Res_Sec_Attr	Provide management on resource security attributes.	T.User_Misuse_Avl_Resc.User_Obst_Res_Use
O.Manage_TSF_Data	Manage security-critical (TSF) data to ensure that the size of the data does not exceed the space allocated for storage of the data.	T.Hack_Avl_Resource.Hack_Stg_Overload T.User_Misuse_Avl_Resc.User_ErrAvl_AudExhst
O.No_Residual_Info	Ensure there is no "object reuse;" i.e., ensure that there is no residual information in some information containers or system resources upon their reallocation to different users.	T.Dev_Flawed_Code.Dev_FC_Buff_Not_Clr T.User_Collect.User_Collect_Residue
O.NonRepud_Assess_Recd	Support nonrepudiation for received information by supporting remote handling of nonrepudiation evidence if needed.	T.Repudiate_Transact.Repudiate_Trans
O.NonRepud_Assess_Sent	Support nonrepudiation for sent information by supporting remote handling of nonrepudiation evidence if needed.	T.Repudiate_Send.Repudiate_Send T.Repudiate_Transact.Repudiate_Trans
O.NonRepud_Gen_Recd	Prevent a receiving user from avoiding accountability for receiving a message by providing evidence that the user received the message.	T.Repudiate_Transact.Repudiate_Trans T.Repudiate_Receive.Repudiate_Rcvr
O.NonRepud_Gen_Sent	Prevent a user from avoiding accountability for sending a message to a recipient at a different site by providing evidence that the user sent the	T.Repudiate_Send.Repudiate_Send T.Repudiate_Transact.Repudiate

	message.	_Trans
O.NonRepud_Locals_Rcvd	Prevent user from avoiding accountability for receiving a message from another user on the same system by providing evidence that the user received the message.	T.Repudiate_Receive.Repudiate_Rcvr_Int T.Repudiate_Transact.Repudiate_Trans
O.NonRepud_Locals_Sent	Prevent user from avoiding accountability for sending a message to another user on the same system by providing evidence that the user sent the message.	T.Repudiate_Transact.Repudiate_Trans
O.NonRepudiate_Recd	Provide evidence that a user received information.	
O.NonRepudiate_Sent	Provide evidence that a user sent information.	T.Spoofing.Hack_Spoof_MsgHdr
O.Obj_Attr_Integrity	Maintain object security attributes with moderate to high accuracy (under the guidance of qualified users).	T.User_Err_Slf_Protect.User_Err_Object_Attr
O.Obj_Protection	Require domain protection for objects. Specify object classes (domains), user groups, and operation classes. Use these to specify which operations may be performed on which objects by which users. Basically this controls what users can do in a given group.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_TSFCode T.Malicious_Code.Mal_Code_Hack_Downld T.Malicious_Code.Mal_Code_IT_Download T.Malicious_Code.Mal_Code_Usr_Downld
O.Permit_Aliases	Permit some users to maintain partial anonymity when using specified services or resources by means of aliases.	T.Admin_UserPriv.Admin_UserPriv_Gen
O.Permit_Anonymity	Permit some users to maintain anonymity when using specified services or resources.	T.Admin_UserPriv.Admin_UserPriv_Gen
O.Prevent_AskPrivInfo	Provide some services or resources to specified users without soliciting from the user information that is relevant to the user's privacy.	T.Admin_UserPriv.Admin_UserPriv_Col
O.Prevent_Link	Ensure that a user may make multiple uses of a service or resource without other specified users being able to link these uses together.	T.Admin_UserPriv.Admin_UserPriv_Agg
O.Prevent_Observe	Ensure that a user may use a service or resource without other specified users being able to observe that the service or resource is being used.	T.Admin_UserPriv.Admin_UserPriv_Agg
O.Priority_Of_Service	Control access to resources so that lower-priority activities do not unduly interfere with or delay higher-priority activities.	T.User_Misuse_Avl_Resc.User_Prcsr_Overload T.Hack_Avl_Resource.Hack_Prcsr_Overload T.Component_Failure.Phys_CompFail_Res T.User_Misuse_Avl_Resc.User_Obst_Res_Use T.Admin_Err_Commit.Admin_Err_Info T.Hack_Avl_Resource.Hack_Comm_Overload T.User_Misuse_Avl_Resc.User_Comm_Overload T.Hack_Avl_Resource.Hack_Stg_Overload

		T.User_Misuse_Avl_Resc.User_Stg_Overload
O.Prvlg_IF_Status	Provide capability for an administrator to determine the use status of all privileged interfaces. This would include interfaces used by maintenance personnel.	T.Admin_Err_Omit.Admin_Err_Omit_Trap
O.Rcv_MsgMod_ID	The TSF recognizes changes to messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.	T.Hack_Msg_Data.Hack_MsgData_RcvUsr
O.Rcv_MsgMod_Rcvr	The TSF detects and corrects changes in messages received from a remote trusted site.	T.Hack_Msg_Data.Hack_MsgData_RcvUsr
O.React_Discovered_Atk	Implement automated notification or other reactions to the TSF-discovered attacks in an effort to identify attacks and to create an attack deterrent.	T.Hack_Avl_Resource.Hack_Prcsr_Overload
O.Reference_Monitor	Always invoke mechanisms that enforce security policies (i.e., as for a traditional reference monitor).	T.User_Modify.User_Modify_TSFData
O.Remote_Execution	Disable a remote entity's ability to execute local code.	T.Malicious_Code.Mal_Code_Hack_Exe T.Malicious_Code.Mal_Code_Hack_Downld
O.Resource_Quotas	Use resource quotas to limit user and service use of system resources to a level that will prevent degradation or denial of service to other critical users and services.	T.Hack_Avl_Resource.Hack_Comm_Overload T.Admin_Err_Commit.Admin_Err_Info T.Hack_Avl_Resource.Hack_Stg_Overload T.Hack_Avl_Resource.Hack_Prcsr_Overload T.User_Misuse_Avl_Resc.User_Stg_Overload T.User_Misuse_Avl_Resc.User_Prcsr_Overload T.Component_Failure.Phys_CompFail_Res T.User_Misuse_Avl_Resc.User_Comm_Overload
O.Robust_Encryption	Produce cipher text that cannot be decrypted without either massive computational power or knowledge of the encryption key through robust encryption techniques.	T.Hack_Crypto.Hack_Crypto_PlnTxt T.Hack_Crypto.Hack_Crypto_ChsnCy T.Hack_Crypto.Hack_Crypto_ChsnPln T.Hack_Crypto.Hack_Crypto_Cypher T.Hack_Crypto.Hack_Crypto_ChsnTxt
O.Rollback	Recover from user operations by undoing some user operations (i.e., "rolling back") to restore a previous known state.	T.User_Err_Inaccess.User_Err_Delete
O.Screen_Lock	Provide a screen lock function to prevent an unauthorized user from using an unattended computer where a valid user has an active session.	T.Hack_Masq.Hack_Masq_Uwkstn

O.Secure_Configuration	Manage and update system security policy data and enforcement functions, and other security-relevant configuration data, in accordance with organizational security policies.	T.Admin_Err_Omit.Admin_Err_Update
O.Secure_State	Maintain and recover to a secure state without security compromise after system error or other interruption of system operation.	T.Dev_Flawed_Code.Dev_FC_Recovery T.Component_Failure.TSF_Err_Conf_Crypto
O.Security_Attr_Mgt	Manage the initialization of, values for, and allowable operations on security attributes.	T.Admin_Err_Commit.Admin_Err_AC_Policy T.User_Err_Inaccess.User_Err_Set_Attr T.Admin_Err_Commit.Admin_Err_User_Attr T.User_Err_Inaccess.User_Err_Mod_Attr T.Admin_Err_Commit.Admin_Err_Resource
O.Security_Data_Mgt	Manage the initialization of, limits on, and allowable operations on security-critical data.	T.Admin_Err_Commit.Admin_Err_AC_Policy T.Admin_Err_Commit.Admin_Err_Sys_Entry T.Admin_Err_Commit.Admin_Err_Authentic T.Admin_Err_Commit.Admin_Err_User_Attr T.Admin_Err_Commit.Admin_Err_Audit T.User_Modify.User_Modify_Auth T.Admin_Err_Commit.Admin_Err_Info
O.Security_Func_Mgt	Provide management mechanisms for security mechanisms.	T.Admin_Err_Commit.Admin_Err_User_Attr T.Admin_Err_Commit.Admin_Err_AC_Policy
O.Security_Roles	Maintain security-relevant roles and the association of users with those roles.	T.User_Modify.User_Modify_Audit T.Admin_Err_Commit.Admin_Err_AC_Policy T.User_Collect.User_Collect_Eaves T.Admin_Err_Commit.Admin_Err_Audit T.Admin_Err_Commit.Admin_Err_User_Attr T.Admin_Err_Commit.Admin_Err_Info
O.Session_Termination	System terminates a session after a given interval of inactivity.	T.Hack_Masq.Hack_Masq_Uwkstn
O.Snt_MsgMod_ID	The TSF supports recognition of changes to transmitted messages that occurred in transit, including insertion of spurious messages and deletion or replay of legitimate messages.	T.Hack_Msg_Data.Hack_MsgData_SndUsr
O.Snt_MsgMod_Rcvr	The TSF supports detection and correction changes in messages sent to a remote trusted site.	T.Hack_Msg_Data.Hack_MsgData_SndUsr

O.Source_Code_Exam	Examine for accidental or deliberate flaws in code made by the developer. The accidental flaws could be lack of engineering detail or bad design. Where the deliberate flaws would include building trapdoors for later entry as an example.	T.Dev_Flawed_Code.Dev_FC_Trap_Door
O.Standard_Output_Pres	Present each possible output value in a standard form.	T.User_Abuse_Conf.User_Abuse_Conf_Steg T.User_Send.User_Abuse_Conf_Steg
O.Storage_Integrity	Provide integrity for data.	
O.Sys_Access_Banners	Inform the user of the possibility of the system monitoring his actions, and that misuse of the system may result in criminal or civil penalties.	
O.Sys_Assur_HW/SW/FW	Ensure that security-relevant software, hardware, and firmware are correctly functioning through features and procedures.	
O.Sys_Backup_Procs	Provide backup procedures to ensure that the system can be reconstructed.	
O.Sys_Backup_Restore	Provide through frequent backups, restoration of security-relevant changes to the system between backup and restore, and restoration of the security-relevant system state (e.g. access control list) without destruction of other system data.	
O.Sys_Backup_Storage	Provide sufficient backup storage and effective restoration to ensure that the system can be recreated.	
O.Sys_Backup_Verify	Detect modifications to backup hardware, firmware, and software.	
O.Sys_Self_Protection	Protect the system security functions through technical features.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_TSFCode T.Dev_Flawed_Code.Dev_FC_Self_Protect
O.Tamper_ID	Provide system features that detect physical tampering of a system component, and use those features to limit security breaches.	T.Hack_Phys.Hack_Phys_Crypto T.User_Misuse_Avl_Resc.User_Obst_Res_Use T.Hack_Phys.Hack_Phys_Damage T.Hack_Comm_Eavesdrop.Hack_CommEaves_Tap
O.Tamper_Resistance	Prevent or resist physical tampering with specified system devices and components.	T.Hack_Phys.Hack_Phys_Crypto T.Hack_Phys.Hack_Phys_Damage T.Hack_Comm_Eavesdrop.Hack_CommEaves_Tap T.User_Misuse_Avl_Resc.User_Obst_Res_Use
O.Trusted_DS_Recov	Ensure that a replaced failed component when re-integrated into the system will recover such that it will not cause errors or security breaches in other parts of the system.	T.Failure_DS_Comp.Failure_DS_Comm
O.Trusted_Path	Provide a trusted path between the user and the	T.Spoofing.Hack_Spoof_Login

	system. Execution of a user-requested action must be made via a trusted path with the following properties: * The path is logically distinct from, and cannot be confused with other communication paths (by either the user or the system). * The path provides assured identification of its end points.	T.User_Collect.User_Collect_Deceive T.Malicious_Code.Mal_Code_Hack_Exe T.Hack_Masq.Hack_Masq_Uwkstn T.Hack_Masq.Hack_Masq_Hijack T.Hack_AC.Hack_AC_Weak
O.Trusted_Path&Channel	Provide a trusted path to security-critical (TSF) data in which both end points have assured identities. For the remote user, there needs to be a trusted channel as well.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_DataAps T.Malicious_Code.Mal_Code_Hack_Exe
O.Trusted_Recovery	Recovery to a secure state, without security compromise, after a discontinuity of operations.	T.Power_Disrupt.Power_Disrupt_Reset
O.Trusted_Recovery_Doc	Provide trusted recovery to ensure that data cannot be lost or misplaced. Any circumstances which can cause untrusted recovery to be documented with mitigating procedures established.	
O.TSF_Mod_Limit	Limit the malicious modification of security-critical (TSF) code and data to include specific system code to prevent the system security protection capabilities from being diminished or weakened.	T.Admin_Hostile_Modify.Adm_Hstl_Mod_TSFCode
O.TSF_Rcv_Err_ID_Loc	Identification by the system (TOE) of modification of security-critical (TSF) data occurring in transit from a remote trusted site must occur.	T.Hack_Msg_Data.Hack_MsgData_RcvTSF
O.TSF_Rcv_Err_ID_Rem	Identification by the remote site of the modification of security-critical (TSF) data occurring in transit from the remote site must occur.	T.Hack_Msg_Data.Hack_MsgData_RcvTSF
O.TSF_Rcv_Err_Rcvr_Loc	Identification and correction of modification of security-critical (TSF) data occurring in transit from a remote site by the system shall occur.	T.Hack_Msg_Data.Hack_MsgData_RcvTSF
O.TSF_Rcv_Err_Rcvr_Rem	Identification and modification of security-critical (TSF) data occurring in transit to the system (TOE) by a remote trusted site must occur, and the remote site shall be able to recover by transmitting a correct version.	T.Hack_Msg_Data.Hack_MsgData_RcvTSF
O.TSF_Snd_Err_ID_Loc	Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the TSF must occur.	T.Hack_Msg_Data.Hack_MsgData_SndTSF
O.TSF_Snd_Err_ID_Rem	Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the remote site must occur.	T.Hack_Msg_Data.Hack_MsgData_SndTSF
O.TSF_Snd_Err_Rcvr_Loc	Identification of modification of security-critical (TSF) data occurring in transit to a remote site by the remote site must occur, and shall be able to recover by retransmitting a correct version.	T.Hack_Msg_Data.Hack_MsgData_SndTSF
O.TSF_Snd_Err_Rcvr_Rem	Identification and correction of modification of security-critical (TSF) data occurring in transit to the remote site by the remote site must occur.	T.Hack_Msg_Data.Hack_MsgData_SndTSF
O.User_Attributes	Maintain a set of security attributes (which may include group membership, clearance, access	T.Admin_Err_Commit.Admin_Err_User_Attr

	rights, etc.) associated with individual users in addition to user identity.	
O.User_Auth_Enhanced	Execute enhanced measures to ensure that either user authentication data cannot be stolen or when it is stolen, it cannot be used to gain access to the system.	T.Hack_Social_Engineer.Hack_SocEng_Password T.Hack_Masq.Hack_Masq_Wauth T.Spoofing.Hack_Spoof_Login
O.User_Auth_Management	Manage and update user authorization and privilege data in accordance with organizational security and personnel policies.	T.Admin_Err_Omit.Adm_Misconfig_User
O.User_Auth_Multiple	Invoke multiple authentication mechanisms, which will provide confidence that the user is who they say they are.	T.Hack_Masq.Hack_Masq_Wauth
O.User_Conf_Prevention	Prevent unauthorized export of confidential information from the TOE with moderate effectiveness.	T.User_Err_Conf.User_Err_Conf_Exp
O.User_Data_Dial-in	Allow dial-in access through secure mechanisms only.	
O.User_Data_Integrity	Provide appropriate integrity protection for stored user data.	
O.User_Data_Transfer	Provide the ability to have physically protected communications lines, intrusion detection for communications lines, and/or need-to-know isolation for communications lines.	
O.User_Defined_AC	Enforce an access control policy whereby users may determine who may access information they control.	T.User_Collect.User_Collect_Deduce T.User_Collect.User_Collect_Browse T.User_Err_Integrity.User_Modify_Data T.User_Modify.User_Modify_Data T.User_Modify.User_Modify_TSFData
O.User_Guidance	Provide documentation for the general user.	T.Hack_Masq.Hack_Masq_Uwkstn T.Hack_Social_Engineer.Hack_SocEng_SysInfo T.User_Err_Inaccess.User_Err_Set_Attr T.User_Err_Inaccess.User_Err_Mod_Attr T.User_Err_Inaccess.User_Err_Delete

Annex C: Some abbreviations

CC Common Criteria
EAL Evaluation Assurance Level
IT Information Technology
PP Protection Profile
SF Security Function
SFP Security Function Policy
SOF Strength of Function
ST Security Target
TOE Target of Evaluation
TSC TSF Scope of Control
TSF TOE Security Functions
TSFI TSF Interface
TSP TOE Security Policy

Annex D sample (minimalist) XML file for SecGen

```
<?xml version="1.0" standalone="yes" ?>
- <PPDocument>
- <Objectives class="SecGen.ObjectiveRepository">
- <Objective class="SecGen.SecObjective">
<objectivedescription>Design administrative functions in such a way that
administrators do not automatically have access to user objects, except for
necessary exceptions. For an accounts administrator, the necessary exceptions
include allocation and de-allocation of storage. For an audit administrator, the
necessary exceptions include observation of audited actions. In general, the
exceptions tend to be role specific.</objectivedescription>

<objectivecoverage>T.Admin_Hostile_Modify.Adm_Hstl_Mod_Data_AC,T.Admin_
Hostile_Modify.Adm_Hstl_Mod_DataAps,T.Admin_Hostile_Modify.Adm_Hstl_Mo
d_SEP,</objectivecoverage>
<objectivename>O.AC_Admin_Limit</objectivename>
<id>1</id>
</Objective>
</Objectives>
- <Threats class="SecGen.ThreatRepository">
- <Threat class="SecGen.SecThreat">
<id>1</id>
<id_old>Admin_Err_Commit</id_old>
<threatname>T.Admin_Err_Commit.Adm_Err_Crypto</threatname>
<threatdescription>An administrator misconfigures cryptographic functions or
stores plaintext keys in insecure areas.</threatdescription>
</Threat>
</Threats>
- <Policies class="SecGen.PolicyRepository">
- <Policy class="SecGen.SecPolicy">
<policyname>P.Audit_Gen_User</policyname>
<policydescription>Individuals shall be held accountable for their
actions.</policydescription>
<policyrationale />
<policycategory>Accountability</policycategory>
</Policy>
</Policies>
- <Assumptions class="SecGen.SecRepository">
- <Assumption class="SecGen.SecAssumption">
<assumptiondiscussion>Category includes considerations of whether the
administrator is hostile or nice, local or remote.</assumptiondiscussion>
<id>1</id>
<assumptionname>A.Admin_Attitude</assumptionname>
<assumptiondescription>This category of assumptions covers the motives, attitude,
competence, and operations of System Administrator
personnel.</assumptiondescription>
</Assumption>
</Assumptions>
- <ComponentRepository class="SecGen.ComponentRepository">
- <SecComponent class="SecGen.SecComponent">
<componentdescription>Partial CM automation</componentdescription>
<id>ACM_AUT.1</id>
```

<componentname>**1**</componentname>
<familyid>**ACM_AUT**</familyid>
</SecComponent>
</ComponentRepository>
</PPDocument>

Annex E : XSLT transormation file for generating Protection Profile Report

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited with XML Spy v4.2 -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="Windows-1252" />

  <xsl:template match="/">
    <html>
      <body>
        <h1>Generated PP/ST</h1>

        <xsl:call-template name="threats" />

        <xsl:call-template name="policies" />

        <xsl:call-template name="objective" />
      </body>
    </html>
  </xsl:template>

  <xsl:template name="objective">
    <h2>Objectives</h2>

    <table border="1" width="100%">
      <tr bgcolor="#9acd32">
        <th width="100px">Name</th>

        <th width="100px">Description</th>

        <th width="100px">Covers</th>
      </tr>

      <xsl:for-each select="PPDocument/Objectives/Objective">
        <tr>
          <td width="50px">
            <a>
              <xsl:attribute name="name">
                <xsl:value-of select="objectivename" />
              </xsl:attribute>

              <xsl:value-of select="objectivename" />
            </a>
          </td>

          <td width="1000px">
            <xsl:value-of select="objectivedescription" />
          </td>

          <td width="100px">
            <xsl:value-of select="translate(objectivecoverage, ',', ' ')" />
          </td>
        </tr>
      </xsl:for-each>
```

```
  </table>
</xsl:template>

<xsl:template name="threats">
  <h2>Threats</h2>

  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Name</th>

      <th>Description</th>

      <th>Covered by</th>
    </tr>

    <xsl:for-each select="PPDocument/Threats/Threat">
      <tr>
        <xsl:variable name="threatname">
          <xsl:value-of select="threatname" />
        </xsl:variable>

        <td>
          <xsl:if
test="count(//PPDocument/Objectives/Objective[contains(objectivecoverage,
$threatname)]) = 0">
            <font color='Red'>!</font>
          </xsl:if>

          <xsl:value-of select="threatname" />
        </td>

        <td>
          <xsl:if
test="count(//PPDocument/Objectives/Objective[contains(objectivecoverage,
$threatname)]) = 0">
            <font color='Red'>!</font>
          </xsl:if>

          <xsl:value-of select="threatdescription" />
        </td>

        <td>
          <xsl:if
test="count(//PPDocument/Oobjectives/Objective[contains(objectivecoverage,
$threatname)]) = 0">
            <font color='Red'>
            </font>
          </xsl:if>

          <xsl:for-each
select="//PPDocument/Objectives/Objective[contains(objectivecoverage,
$threatname)]">
            <a>
              <xsl:attribute name="href">#
              <xsl:value-of select="objectivename" />
              </xsl:attribute>
```

```
                <xsl:value-of select="objectivename" />
              </a>

              <br />
            </xsl:for-each>

            <xsl:if
test="count(//PPDocument/Objectives/Objective[contains(objectivecoverage,
$threatname)]) = 0">
              <font color='Red'>ERROR: UNCOVERED THREAT!</font>
            </xsl:if>
          </td>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>

  <xsl:template name="policies">
    <h2>Policies</h2>

    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Name</th>

        <th>Description</th>

        <th>Discussion</th>

        <th>Covered by</th>
      </tr>

      <xsl:for-each select="PPDocument/Policies/Policy">
        <tr>
          <td>
          <xsl:value-of select="policyname" /></td>
          <td>
          <xsl:value-of select="policydescription" />
           </td>

          <td>
          <xsl:value-of select="policydiscussion" />
                  </td>
                  <td>
          <xsl:variable name="name">
          <xsl:value-of select="policyname" />
          </xsl:variable>
          <xsl:for-each
select="//PPDocument/Objectives/Objective[contains(objectivecoverage, $name)]">
            <a><xsl:attribute name="href">#
              <xsl:value-of select="objectivename" />
              </xsl:attribute><xsl:value-of select="objectivename"
/></a><br/></xsl:for-each></td>
        </tr>
        </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>
```